The Report Committee for Bhargavi Narayanasetty
Certifies that this is the approved version of the following report

Analysis of High Performance Interconnect in SoC with distributed Switches and Multiple Issue Bus Protocols

APPROVED BY
SUPERVISING COMMITTEE:

Supervisor:

Lizy Kurian John

Steve Korson

Analysis of High Performance Interconnect in SoC with distributed Switches and Multiple Issue Bus Protocols

by

Bhargavi Narayanasetty, BTech

Report

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Engineering

The University of Texas at Austin

May 2011

Acknowledgements

Foremost, I would like to thank my Project Supervisors, Prof. Lizy Kurain John and Steve Korson, who shared with me a lot of their expertise and insight on so many aspects of this project. With his enthusiasm, inspiration, and great efforts to explain things clearly and simply, Steve, helped to make this challenging project an amazing learning experience. Through all my interactions, Prof. Lizy, provided encouragement, sound advice, and shared lots of great ideas and valuable knowledge. She has believed in me from day one and guided me through every step of my studies, I am grateful for her unconditional support. I really would have been lost, without such amazing mentors.

I would also like to express my gratitude to Prof. Yale Patt, his academic guidance and encouragement provided a sense of direction throughout the course of studies here at UT-Austin. Many thanks, to my team with the ASIC-SoC group at Texas Instruments. Their thoughtful advice and co-operation helped me through this unique experience within a challenging technology development environment.

I am deeply grateful to the Department of Electronics & Computer Engineering at The University of Texas at Austin for such a great learning experience, a flexible curriculum and inspiring teachers. I am also thankful to Texas Instruments for providing me this opportunity to work in the co-op program over Summer and Fall 2010. I am sure to carry the experience and knowledge gained over the past two years for the near and far future and hope to build positively upon that.

Special thanks, to my family and friends for their support and kindness.

May 2011

Abstract

Analysis of High Performance Interconnect in SoC with distributed Switches and Multiple Issue Bus Protocols

by

Bhargavi Narayanasetty, M.S.E

The University of Texas at Austin, 2011

Supervisor: Lizy K John

In a System on a Chip (SoC), interconnect is the factor limiting Performance, Power, Area and Schedule (PPAS). Distributed crossbar switches also called as Switching Central Resources (SCR) are often used to implement high performance interconnect in a SoC – Network on a Chip (NoC). Multiple issue bus protocols like AXI (from ARM), VBUSM (from TI) are used in paths critical to the performance of the whole chip. Experimental analysis of effects on PPAS by architectural modifications to the SCRs is carried out, using synthesis tools and Texas Instruments (TI) in house power estimation tools. The effects of scaling of SCR sizes are discussed in this report. These results provide a quick means of estimation for architectural changes in the early design phase. Apart from SCR design, the other major domain, which is a concern, is deadlocks. Deadlocks are situations where the network resources are suspended waiting for each other. In this report various kinds of deadlocks are classified and their respective

mitigations in such networks are provided. These analyses are necessary to qualify distributed SCR interconnect, which uses multiple issue protocols, across all scenarios of transactions. The entire analysis in this report is carried out using a flagship product of Texas Instruments. This ASIC SoC is a complex wireless base station developed in 2010-2011, having 20 major cores. Since the parameters of crossbar switches with multiple issue bus protocols are commonly used in SoCs across the semiconductor industry, this reports provides us a strong basis for architectural/design selection and validation of all such high performance device interconnects.

This report can be used as a seed for the development of an interface tool for architects. For a given architecture, the tool suggests architectural modifications, and reports deadlock situations. This new tool will aid architects to close design problems and bring provide a competitive specification very early in the design cycle. A working algorithm for the tool development is included in this report.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

The growth of Moore's law has shown way for many applications that were impossible to manage humanely just a decade ago. The increasing computational capabilities and ability to enable integrated systems as intelligent components has helped us manage the ever-increasing flow of data and processes within the system as well as over a network. We have a vast variety of complex systems, right from the general purpose CPU units that power everyday computers to the graphics cores to the many more mobile processors and application specific processors that are designed to perform certain functionality with extreme reliability.

Memory and computational capabilities are two of the most ephemeral constraints for today's computing systems. Integrated solutions for scalable performance needs are becoming more and more commonplace. In the current era of digital electronic systems, there is a growing need for hardware to meet up to requirements of computationally intensive applications. From the point of view of a technology company, competitiveness is not only for providing the solutions, but also meeting the Performance, Power, Area and Schedule (PPAS). With the portability factor and green computing becoming active concerns for all of the technology needs, it is imperative to focus on the main parameters that relate the viability, efficiency and performance of a device coming out in the market early in the design phase. The concept of System on a Chip (SoC) has revolutionized the industry because it encompasses multiple processing units on a single chip. The idea is to have many computational resources (cores) such as CPUs, specialized DSP processors and memory units on a single chip. The current trends in VLSI technology have made this possible with the capability of billions of transistors on a single chip.

MOTIVATION AND OVERVIEW

The goal of SoC is to produce high performance units with a reduction in the cost and development time by assembling a set of working components efficiently. The systems methodologies of modular design and reuse of complex components are utilized to achieve this with high throughput.

High performance and low energy consumption of these SoCs, depends mainly on an efficient integration of these cores, through the on-chip interconnect. In the given architecture, on-chip communication within many processing units is now becoming the bottleneck. The factor limiting PPAS in a system is the movement of data. The challenge is to design an efficient interconnect, which gives the required performance making an efficient use of the resources providing high bandwidth, low latency communication between cores with a minimum cost and energy.

Long the mainstay of on-chip network interconnections consisting of conventional bus or cross bar structures inspired from the network topologies. But in such topologies, the interface for all the interconnecting cores should be identical in terms of the design parameters (clock frequency and timing margins) and logically (signal). This turns out to be a significant obstacle in the rapid integration and reuse of existing components. This integration challenge was addressed by the approach usually called Network on a Chip (NoC) or On Chip Networks or Interconnect Fabrics and so on. We will use Network on a chip as an umbrella name where the actual component and Interconnect are decoupled. In effect, it incorporated the bus related logic into interconnect, instead of implementing it in the components [19, 22].

Due to NoC, it is possible to easily integrate multiple components with different Clock Domains, Bus Widths, Power Domains, without changing the components themselves. The concept of NoC helps industry to integrate various components from

different vendors with out any dependencies on their respective architectures. Hence it provides a two-fold development, one – There is always a flexibility to take the best in class component and two – due to the competition across vendors there is an ever-increasing innovation.

The Network on a Chip (NoC) includes components in charge of directing the ongoing traffic toward the corresponding destinations. These components are usually called as the "Routers", "Switch Fabrics", "Switches", "Switching Central Resources (SCR)" (terminology used by Texas Instruments (TI)). The architecture may also use components to connect segments running at different frequencies, different data-path widths or different communication bus protocols. These are generally referred to as the Bridges.

Overview of Switching Central Resource (SCR) in NoC

Switching Central Resources (SCR) [1] mainly consists of a switch matrix, network interfaces with a communication protocol and routing and arbitration techniques. The main architectural and design parameters of the SCRs include [1, 9, 10, 11]

- Number of Masters and Slaves (number cores initiating requests and number of cores servicing them)
- The switching interconnect (Switching Matrix) topologies such as Crossbar, Bus, Ring, Star, Multi-stage Interconnect Network (MIN). Currently in SoC, Crossbar and MIN-static are dominant topologies, as the IP cores on chip are limited to a maximum of 16 to 64. As these IP cores increase, a greater variety of network implementations are expected.

- The standard bus protocol to use such as AXI (from ARM), VBUSP and VBUSM (from Texas Instruments), OCP (from OCP-IP)
- Bus Width
- The number of pipeline stages
- The Voltage Domain
- The frequency of implementation
- The arbitration scheme such as fixed priority, first come first serve, round robin, hybrid

The choice of these architectural and design parameters have a large impact on the performance, power dissipation and area of the Bus Architecture and hence the SOC. With the increase in the number of IP cores communicating, the sizes of the SCRs are increasing.

Overview of Deadlocks

Deadlock is a condition in which the throughput of a network or part of a network goes to zero due to conflicts in resource acquisition. In other words, a network is in deadlock if the network resources are suspended waiting for each other. There does exist work regarding the analysis and solutions to avoid the various kinds of deadlocks in networks, which consist of a single SCR [5, 6].

In the current complex SoC Bus Architectures, multiple SCRs are interconnected with or without bridges to obtain the required performance and cost metrics. This is a distributed SCR network [8, 12]. Figure 1 is an example of a Distributed SCR network. In this example there are 4 CPU processing units (CPU-0, CPU-1, CPU-2, CPU-3), 4 DSP processing units (DSP-0, DSP-1, DSP-2, DSP-3), 2 Local Memories (L-Mem0, L-Mem1)

and 1 Global Memory (G-Mem). The Bus Architecture in this case consists of 5 SCR units.

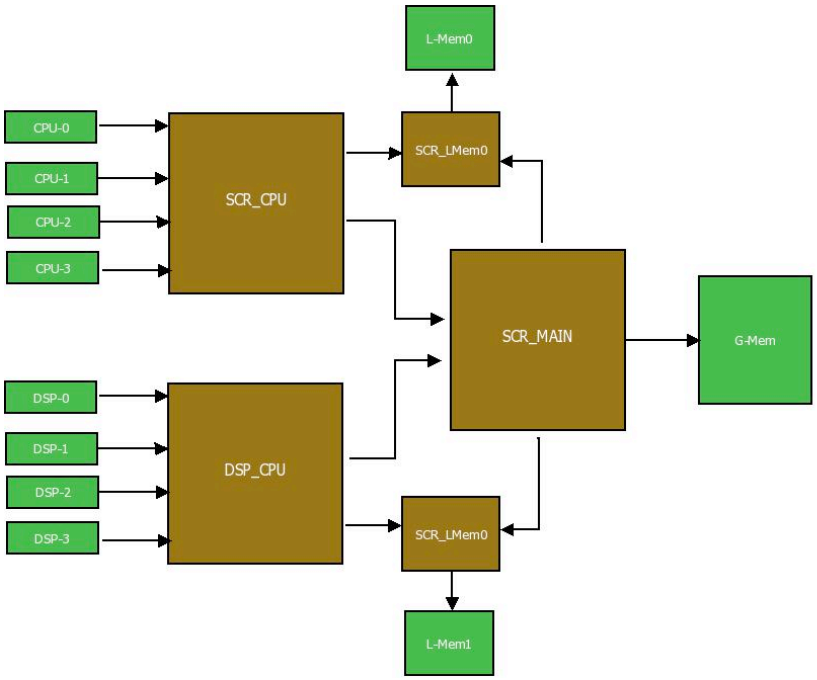

Figure 1: Distributed SCR network

In multiple issue bus protocols such as the VBUSM protocol [1] or AXI protocol [3], it allows multiple transactions to be outstanding, implemented by transaction pipelining which drastically increases the system performance. In such protocols, the master sends out multiple write requests (address requests), but sends the write data in the same order of the requests, after the slave has acknowledged the previous request.

In a distributed network of SCRs that use multiple issue protocols such as the VBUSM, new kinds of deadlocks occur. One such deadlock class in if there are parallel paths to a single slave. A deadlock of this class is a mechanism where it's possible for two write-commands from the master, to take different paths to the same slave. In such

cases its possible for the second command to surpass the first. If this occurs the slave sends an acknowledgment for the second request and is waiting for the second requests data, where as the master is waiting for the first requests acknowledgment, which is not received and hence does not send the second request data creating a deadlock.

OBJECTIVE OF THE REPORT

There are two main sections of this report. Together they help design and analyze a high performance interconnect to meet the requirements in a SOC specification. The entire analysis in this report is carried out using a flagship product of Texas Instruments. This ASIC SoC is a complex wireless base station developed in 2010-2011, having 20 major cores. Since the parameters of crossbar switches with multiple issue bus protocols are commonly used in SoCs across the semiconductor industry, this reports provides us a strong basis for architectural/design selection and validation of all such high performance device interconnects.

The first section of this report deals with the effects of the architectural design choices on the performance and cost metrics namely the latency, area and power of the SCR. The analysis is carried out using a Common Bus Architecture [1] (developed by Texas Instruments), which uses a cross bar switching matrix and VBUSP and VBUSM protocols (similar to AXI by ARM) to communicate. As the frequency of implementation, bus width and the bus protocol are a part of the specifications of any design, the variations with respect to number of masters, number of slaves, arbitration schemes (round robin, fixed priority and dynamic round robin fixed priority), voltage domain, number of pipeline stages are analyzed. In addition a case analysis of the main SCR used by a Texas Instruments 20 core wireless base station is carried out and reported. These experimental analyses provides a means of high level estimations and

analysis of SCR in the early stages of the design process, which helps avoid re-design in the middle of the design cycle which leads to an increase in the time to market. Since these parameters are currently most widely used (switch network of crossbars [2] and AXI similar multiple address issue bus protocols [3, 4]) in SoCs, these analysis results would be useful estimates for many similar SCRs.

The second session of this report, deals with the classification and analysis of the various kinds of deadlocks, which can occur in bus architectures with distributed network SCRs and similar multiple issue protocols. This analysis is a necessity to design such complex bus architectures [7, 8] to be functional without the issue of deadlocks. It also deals with a case analysis of the bus architecture design of a Texas Instruments 20 core wireless base station.

While resolving deadlocks, one may need to modify the architecture of interconnect. Due to these changes the Power, Performance and Area (PPA) will be impacted. Any new changes required to meet the PPA specification can lead to unexpected deadlocks. Hence a vicious circle between PPA estimation and deadlock analysis exists.

This report can be used as a seed for the development of an interface tool for architects. For a given architecture, the tool suggests architectural modifications, and reports deadlock situations. This new tool will aid architects to close design problems and provide a competitive specification very early in the design cycle. A working algorithm for the tool development is included in this report.

ORGANIZATION

The remainder of this report is organized as follows.

- Chapter 2 provides describes the evolution of Interconnect Bus Architectures in a Soc as well as discusses other existing architectures (across other semiconductor players in the industry).
- Chapter 3 will present the overview of the NoC case study, providing all the terminologies, which are used in this report.
- Chapter 4 will present the analysis of SCR performance and cost metrics with changes in architectural parameters.
- Chapter 5 will present the Case Analysis of the main SCR in a TI Wireless Base Station device, developed for a major client in the year 2010-2011.
- Chapter 6 will present and discusses the various kinds of deadlocks, which can occur in distributed networks with multiple issue protocols.
- Chapter 7 will present an algorithm to detect deadlocks in a distributed SCR Network. It also includes the case analysis of the Bus Architecture of a Texas Instruments Wireless base station.
- Finally the conclusion can be found in Chapter 8

Chapter 2: Evolution of SoC Interconnect Architectures

Complex systems have always been hard to build and debug. SoCs are no different in this respect, but the economics of integrated circuit manufacture do not allow debug by trial and error. The design must be right the first time. As a result SoC designers use methodologies to minimize risk of design error. One such methodology is to be systematic about the way interconnect is used in a chip. Any communication failure, whether due to noise or an error in timing or protocol, is likely to require a design iteration that will be expensive in both mask charges and time to market.

The design of these interconnects has evolved ever since the first SoC, from the long mainstay of basic bus or crossbar approach to the Network on a Chip (NoC). In this chapter we discuss the various SoC Interconnect Architectures, which evolved over the years of design development. SoCs typically consist of several complex heterogeneous components (‘nodes’), such as programmable processors, dedicated (custom) hardware to perform specific tasks, on-chip memories and input–output interfaces. The on-chip communication architecture serves as the interconnection fabric for communication between these components. The concepts of the complex NoC architectures are discussed here starting from the basic ‘Single Bus’ architecture.

SINGLE BUS ARCHITECTURE

The early SoC interconnect prototype was inspired by the Rack-based [21] microprocessors of the earlier age. In those rack systems, a backplane of parallel connections formed a ‘bus' into which all manner of cards could be plugged. A system designer could select cards from a catalogue and simply plug them into the rack to yield a

customized system with the processor, memory and interfaces required for any given application.

In a similar way, a designer of an early SoC could select nodes, place them onto the silicon, and connect them together with a standard on chip bus (see Figure 2). The backplane might not be apparent as a set of parallel wires on the chip, but logically the solution is the same.

In a 'Bus', the interconnect is mostly just wires, interconnecting nodes, combined with an arbiter that manages access to the 'Bus'.

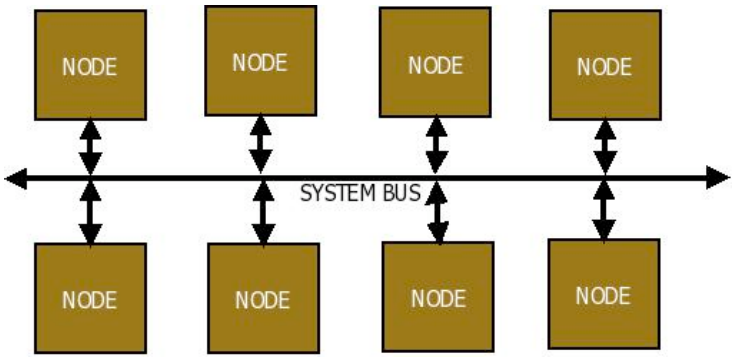

Figure 2: Single Bus (CIRCA 1995, interconnect as a Single Bus in SoC).

Advantages

1. Low Complexity in Design, easy to implement and extend.
2. Less Area (No additional modules in the interconnect).
3. Well suited if the number of nodes is small, which do not require high speeds (fast in such cases).
4. Easy identification of faults.

Disadvantages

1. Suffers from power scalability. Increase in the number of nodes, increases the capacitive load on the bus.
2. Performance degrades as the number of nodes increase, works best with a limited number of nodes.
3. If there is a problem anywhere in the bus, the entire interconnect fails.
4. Commonly has lower bandwidth compared to other architectures.
5. Only one communication between nodes can occur at any point in time. It can easily become a communication bottleneck.
6. High priority accesses are stalled by transactions in progress.
7. Bandwidth is limited by clock frequency, which itself is limited by the physical design parameters such as length of the wires

POINT TO POINT ARCHITECTURE

Here each component/node has a dedicated connection towards other dependent components/nodes (Figure 3). In the figure, Node 1 has dependencies to Node 3 and 4. Node 2 has dependencies with Node 4, and Node 3 has dependencies with Node 1 and Node 4.

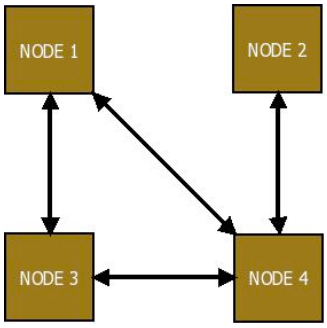

Figure 3: Point to Point Architecture

The practical implementations of a point-to-point network adopt orthogonal topologies[23], where nodes can be arranged in an n dimensional orthogonal space; in such a case every connection produces a displacement in a single dimension. Control of message parts (i.e. routing) in these networks is simple and can be implemented efficiently in hardware. Examples of popular orthogonal networks are n-dimensional Mesh, Torus and the Hypercube.

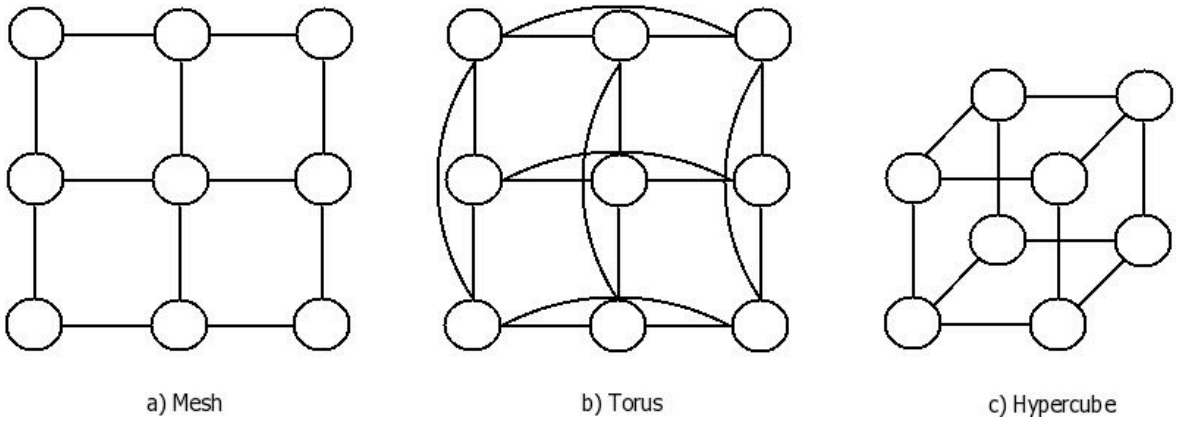

Figure 4: Examples of Orthogonal Networks

Advantages

1. Multiple communications between modules can occur simultaneously.
2. Easier to detect faults, a single fault does not disconnect the communications between the other nodes.
3. Well suited if the number of nodes is small, which do not require high speeds (fast in such cases).

Disadvantages

1. Adding a node may require modifications to all the nodes communicating with the new node (to add the port to communicate).
2. Suffers from power scalability. Increase in the number of nodes, increases the capacitive load on the bus.
3. Performance degrades as the number of nodes increase, due to the increase in capacitance. Works best with a limited number of nodes.
4. Expensive realization costs in terms of chip area.

CROSSBAR SWITCH ARCHITECTURE

With the limitations of the above mentioned architectures, and faced with increasing IC performance requirements, designers started to implement cross-bar structures (Figure 5). These structures improve latency predictability and significantly increase aggregate bandwidth, at a not-insignificant cost, of course, of a much larger number of wires.

A crossbar is a switch connecting multiple inputs to multiple outputs in a matrix manner.

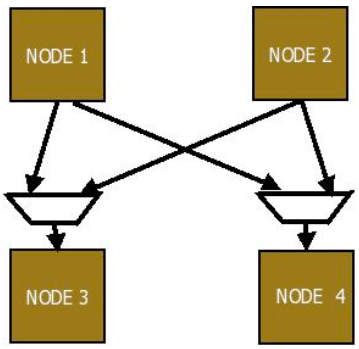

Figure 5: Cross-bar Switch Architecture

Advantages

1. Improved latency predictability.
2. Increase in aggregate band-width
3. Parallelisms in terms of one-input sources sends to multiple output sources
4. Multiple input sources sending to multiple output sources without blocking each other.

Disadvantages

1. Does not scale efficiently in terms of area and power
2. Expensive realization costs in terms of chip area
3. Complex wire routing with more number of nodes, which introduces larger power consumption and interconnect delay

STANDARD BUS PROTOCOLS

In a SoC, several different components such as processors, custom hardware, memories, and external interface hardware need to communicate with each other. Some of these components are re-used from previous designs or obtained for external vendors, where as some components are designed from scratch. In order to efficiently integrate these components, some kind of standard interface definition is essential. Without this standard, the components interface may not be compatible with the bus architecture implementation. Wrappers at the interfaces of components would need to be designed which consume additional area on the SoC and time to design and verify.

Several bus based data transfer protocols were introduced, to speed up the SoC integration and promote re-use of components. These protocols define the number and

the functionality of the pins at the interface. Here we will have a look at some of the commonly used bus protocols.

AMBA: Advanced Microcontroller Bus Architecture

AMBA [3] is used for high performance systems and high-bandwidth interfaces, such as SoCs using the ARM cores. This architecture is capable to cater high external memory bandwidth, on which the CPU, on-chip memory and other Direct Memory Access (DMA) devices reside.

Within AMBA specification, there are three distinct buses:

1. The Advanced High-performance Bus (AHB)
2. The Advanced System Bus (ASB)
3. The Advanced Peripheral Bus (APB)

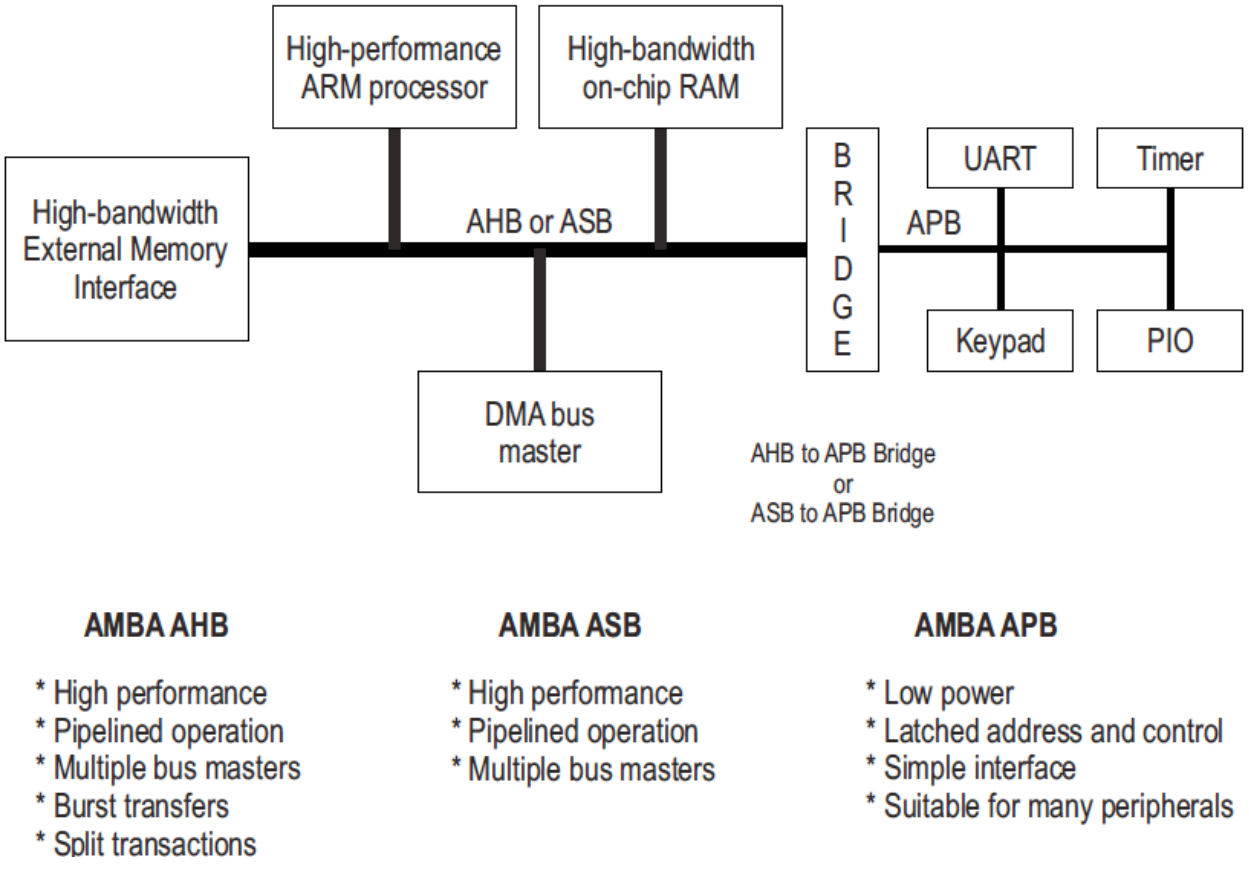

Figure 6: AMBA (Source: ARM Inc. [3])

Similar to Advanced High Performance Bus (AHB) bus protocol, which was introduced in AMBA version 2, Advanced eXtensible Interface (AXI) was introduced in AMBA -3 specification by ARM.

AXI: Advanced extensible Interface

Advanced Microcontroller Bus Architecture – AXI [3], the third generation of AMBA interface defined in the AMBA 3 specification, is targeted at high performance, high clock frequency and includes features to cater for high speed sub-micrometer interconnect.

The main objectives of AXI are to:

1. Handle high-bandwidth and low-latency designs
2. Avoid complex bridges and yet enable high-frequency operation interface wide range of components
3. Be suitable for memory controllers with high initial access latency
4. Be flexible in the implementation of interconnect architectures
5. Be backward-compatible with existing AHB and APB interfaces

The AXI protocol is burst-based. The write data channel to the slave or a read data channel to the master is used for the data to be transferred between master and slave.

The AXI protocol enables:

1. Support for out-of-order completion of transactions
2. Address information to be issued ahead of the actual data transfer (multiple issue protocol)
3. Separate address/control and data phases
4. Support for unaligned data transfers using byte strobes
5. Support for multiple outstanding transactions (multiple issue protocol)
6. Burst based transactions with only start address issued
7. Issuing of multiple outstanding addresses (multiple issue protocol)
8. Easy addition of register stages to provide timing closure

Wishbone

Wishbone bus protocol [24] is an open-source hardware computer bus. This was developed to allow the connection of various cores to communicate inside of a chip. The Wishbone is treated as a “logic bus”, where the specification is written in terms of signals, clock cycles and logic levels.

The advantage of using this architecture is that it aids designers by providing a standard way for hardware logic designs (called "cores" or “nodes”) to be combined. Wishbone is defined to have 8, 16, 32, and 64-bit buses. All signals are synchronous to a single clock but some slave responses must be generated combinatorial for maximum performance.

Wishbone is easily compatible with common topologies such as point-to-point, many-to-many (i.e. the classic bus system), hierarchical, or even switched fabrics such as crossbar switches. In more complex topologies, Wishbone requires a bus controller or arbiter.

OCP: Open Core Protocol

The open core protocol [25] is developed as a part of a non-profit, and open-industry standard body. This bus protocol is highly configurable and scalable interface for on-chip subsystem communications.

The Open Core Protocol

1. Achieves the goal of IP design reuse. The OCP transforms IP cores, making them independent of the architecture and design of the systems in which they are used.

2. Optimizes die area by configuring into the OCP interfaces only those features needed by the communicating cores.
3. Simplifies system verification and testing by providing a firm boundary around each IP core that can be observed, controlled, and validated.
4. Provides tools to validate the OCP interface configuration to ensure rapid creation and integration of interoperable components in industry standard tools

The OCP-IP Specification defines, for each OCP-IP interface, a certain number of parameters. That is by specifying the presence of a signal and width of a signal.

Disadvantages of a Standard Bus Protocol

1. Agreeing upon one all-purpose interconnection protocol, has been unsuccessful due to:
    a. Commercial issues
    b. Disagreement over required features
    c. Different applications require different trade offs
2. Bus protocols limit the reuse of design components.

HIERARCHY OF BUSSES

Due to the scalability limits of the previous architectures, increase in performance demands and increase in the number of nodes to be connected, SoCs could not be built only around a single bus. Instead complex hierarchy of busses [20] (Figure 7), with sophisticated bus protocols and multiple bridges between them evolved.

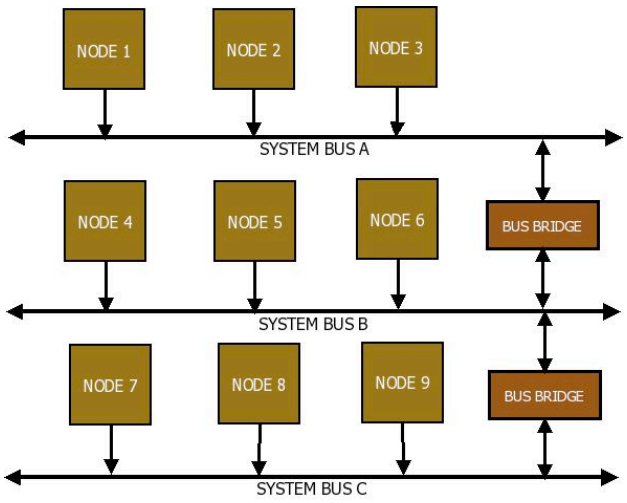

Figure 7: Hierarchy of Busses Architecture

Here two nodes can communicate through multiple busses. The bus bridge here transfers data from one bus to another. In the case where the two busses use different protocols, the bridge does the conversion.

Advantages

1. Faster than a single bus, has lesser load on a single bus
2. Multiple transactions can occur simultaneously if they are in different busses.

Disadvantages

1. Transactions over bridges involve an additional overhead
2. Data needs to be kept, as to which node is on which bus, to allow proper routing.
3. Only a single transaction can occur between nodes on the same bus.

4. During a transfer among nodes in different buses, both busses remain inaccessible to other nodes

NETWORK ON CHIPS (NOC)

As the number of integrated cores (nodes or IPs) is rapidly increasing, traditional interconnects are starting to show scalability issues.

For example in Buses we have a problem of increasing capacitive load. The increasing number of attached cores results in an increased capacitive load, which finally leads to a communication bottleneck.

On the other end of using crossbars or Point to Point interconnect protocols, they do not scale efficiently beyond 20 cores and leads to an increase in area as the number of cores keep increasing.

The previously discussed interconnect architectures even those utilizing hierarchies of buses e.g. high-speed processor bus, system bus and low-speed peripheral bus separated by bridges (similar to the hierarchy of buses offered by ARM); are not be sufficient for many current and future SoCs for several reasons [22, 18]:

1. A (single) bus does not provide concurrent transactions: depending on the arbitration algorithm, access to the bus is granted to that medium/component/device that has the highest priority. All other requests occurring during other current bus transactions have to be postponed to a later point in time. Thus, the bus is blocking other transactions that could potentially executed in parallel.

2. Large bus lengths are prohibitive in a few current and future SoC designs since the combination of large (geometrical area) SoCs and high clock frequencies lead to non-manageable clock skews on a bus-based system.
3. Application convergence: The mixing of various traffic types in the same SoC design (Video, Communication, Computing and etc.). These traffic types, although very different in nature, for example from the Quality of Service point of view, must now share resources that were assumed to be "private" and handcrafted to the particular traffic in previous designs.
4. Moore's law is driving the integration of many design units in a single chip. This is an enabler to application convergence, but also allows entirely new approaches (parallel processing on a chip using many small processors) or simply allows SoCs to process more data streams (such as communication channels)
5. Consequences of silicon process evolutions between generations: Gates cost relatively less than wires, both from an area and performance perspective, than a few years ago.
6. Time-To-Market pressures are driving most designs to make heavy use of synthesizable RTL rather than manual layout, in turn restricting the choice of available implementation solutions to fit a bus architecture into a design flow.
7. In the other previously mentioned architectural topologies, the interface for all the interconnecting nodes should be identical in terms of design parameters (clock frequencies, timing margins) and logically (signals, should have the same protocol for communicating). This is a significant

obstacle for the rapid integration and reuse of existing components which is desired in SoC

Hence the solution to the problems is to adopt an on-chip data-routing network generally known as Network-on-Chip (NoC) architecture. Switched Fabrics [23] is a topology where the nodes connect to each other via one or more switches (usually cross-bar switches). The Switched fabrics form the core of a NoC.

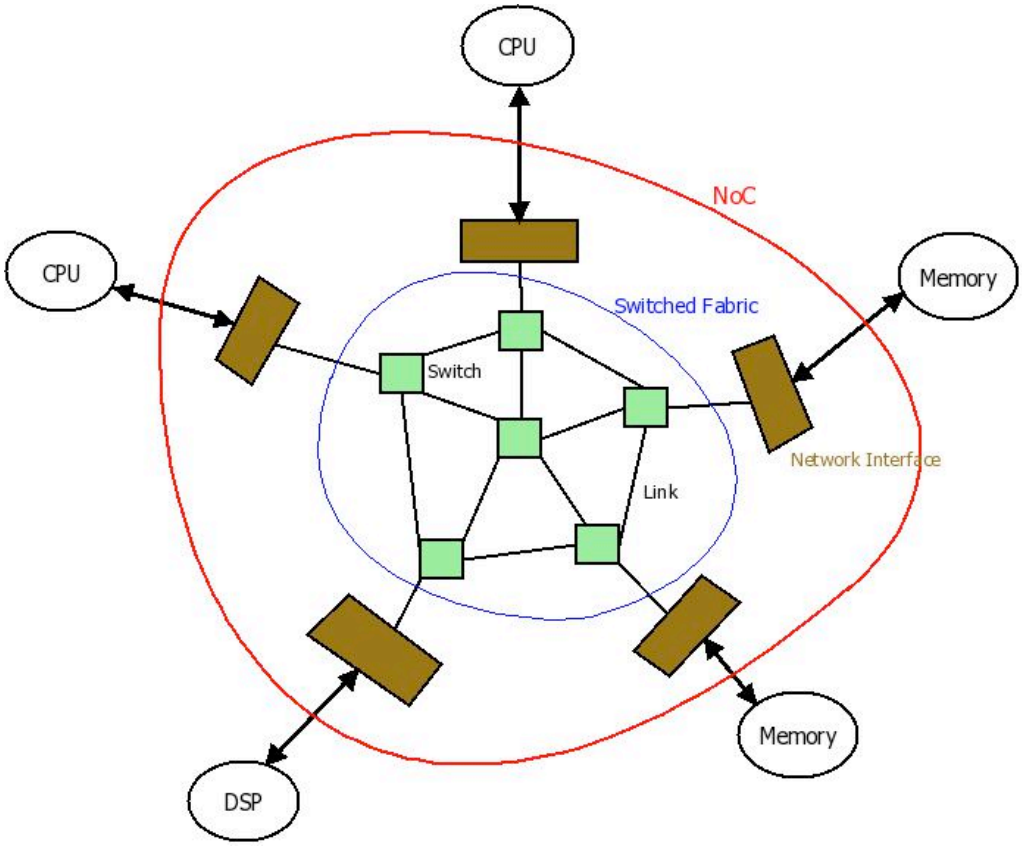

Figure 8: Network on a Chip (NoC), Overview

NoC Layers

The nodes in a SoC communicate over a NoC using a three-layered communication scheme [18, 20, 23] (Figure 9) referred by the transaction, transport and physical layers.

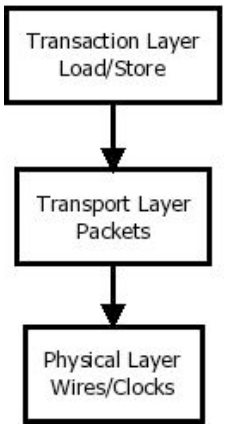

Figure 9: NoC Layers

The Transaction layer defines the communication primitives available to interconnected nodes. The transaction layer defines how information is exchanged between nodes to implement a particular transaction. For example, a NoC transaction is typically made of a request from a Master node to a Slave node, and a response from the Slave to the Master. However, the transaction layer leaves the implementation details of the exchange to the transport and physical layer.

The Transport layer defines rules that apply as packets are routed through the switch fabric. Very little of the information contained within the packet (typically, within the first cell of the packet, a.k.a header cell) is needed to actually transport the packet. The packet format is very flexible and easily accommodates changes at transaction level without impacting transport level. For example, packets can include byte enables, parity

information, or user information depending on the actual application requirements, without altering packet transport, nor physical transport.

A single NoC typically utilizes a fixed packet format that matches the complete set of application requirements. However, in some NoCs consisting of nodes using different packet formats are communicated in the Switched Fabric through translation units (Bridges). The Transport Layer may be optimized to application needs.

The Physical layer defines how packets are physically transmitted over an interface, much like Ethernet defines 10Mb/s, 1Gb/s, and etc. physical interfaces.

Protocol layering allows multiple physical interface types to coexist without compromising the upper layers. Thus, NoC links between switches can be optimized with respect to bandwidth, cost, data integrity, and even off-chip capabilities, without impacting the transport and transaction layers.

A summary of the mapping of the protocol layers into NoC design units is illustrated in Figure 10

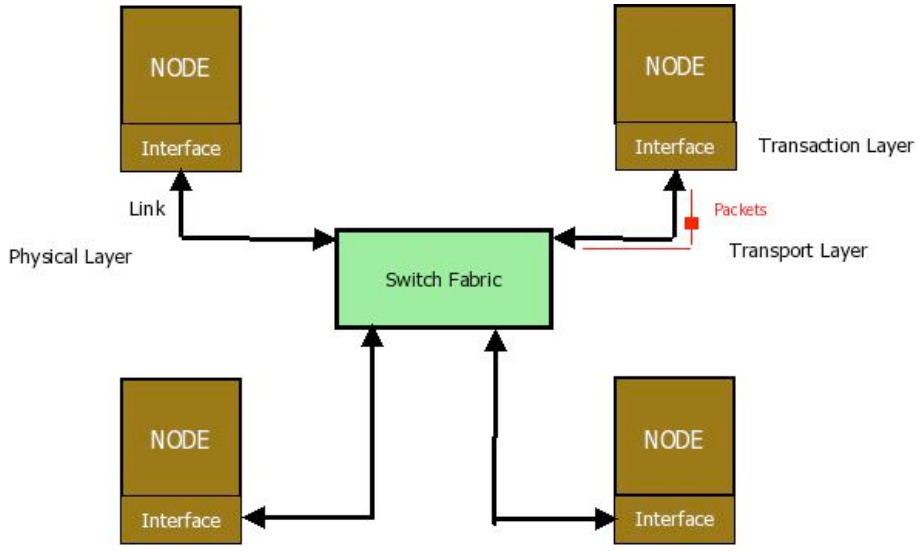

Figure 10: NoC Layers Mapping onto NoC Design Units

NoC Layered Approach Benefits

1. Decoupling the nodes from the Interconnect.
2. Separate optimizations of transaction and physical layers. The transaction layer is mostly influenced by application requirements, while the physical layer is mostly influenced by Silicon process characteristics. Thus the layered architecture enables independent optimization on both sides.
3. Scalability. Since the switch fabric deals only with packet transport, it can handle an unlimited number of simultaneous outstanding transactions (e.g., requests awaiting responses). Conversely, nodes deal with transactions; their outstanding transaction capacity must fit the performance requirements of the block or subsystems that they service.
4. Aggregate throughput. Throughput can be increased on a particular path by choosing the appropriate physical transport, up to even allocating several physical links for a logical path. Because the switch fabric does not store transaction state, aggregate throughput simply scales with the operating frequency, number and width of switches and links between them, or more generally with the switch fabric topology.
5. Quality Of Service. Transport rules allow traffic with specific real-time requirements to be isolated from best-effort traffic. It also allows large data packets to be interrupted by higher priority packets transparently to the transaction layer.
6. Timing convergence. Transaction and Transport layers have no notion of a clock: the clocking scheme is an implementation choice of the physical layer. This enables the NoC to span a SoC containing many IP Blocks or

groups of blocks with completely independent clock domains, reducing the timing convergence constraints during back-end physical design steps.

7. Easier verification. Layering fits naturally into a divide-and-conquer design & verification strategy. For example, major portions of the verification effort need only concern itself with transport level rules since most switch fabric behavior may be verified independent of transaction states. Complex, state-rich verification problems are simplified to the verification of single nodes.
8. Customizability. The Switch fabric can be costumed to the SoC performance needs.
9. As shown by many other analyses [19, 20, 29] NoC solutions lead to a more power efficient solution as compared to the single or hierarchical busses and crossbars.

Limitations of NoC

Scalability and reusability are critical for any on-chip communication platform. NoC proposals are promising but they have several limitations [22].

First, memory access between nodes and memory cores will be a performance bottleneck if a switched fabric has to be traversed on every memory access. Adding local memory resources can ameliorate the problem. However, many data intensive real time applications, especially in media processing, require memory buffers to be shared among many computation cores. Therefore, it may be necessary to evolve a hybrid infrastructure that can provide both circuit switched access to memory cores and allow packet switched data transfer for general data communication.

Second, there are no standards for on-chip networks. On the one hand, absence of standards offers immense opportunities to customize the communication fabric. However, without some standardization, it will be difficult to promote large-scale reusability of the communication backbone and motivate IP vendors to create design independent communication IP cores.

Third, adoption of networks on chips is a slow process. Today, a large majority of designers are not experts in network concepts. Furthermore, unlike IP components for computation, scalable communication IP cores that can be reused across designs are far and few. Therefore, design methodologies and tools that can encapsulate and automate many of the design aspects are necessary to reign in the communication design costs. Models of computation that capture the communication requirements as well as abstract the capabilities of communication architecture can enable efficient matching of application requirements and architectural capabilities. Technologies and tools to perform the match are necessary to reduce the design effort.

Switched Fabrics

As seen above, Switched Fabrics form a major part of a NoC design. Many topologies exist to form Switched Fabric by interconnecting switches. These topologies can be distinguished into 2 basic categories.

1. Regular most common typology: Some examples are (Figure 11)
    a. Mesh
    b. Torus
    c. Binary Tree

2. Irregular derived by mixing different forms in a hierarchical, hybrid, or asymmetric way (clustering)

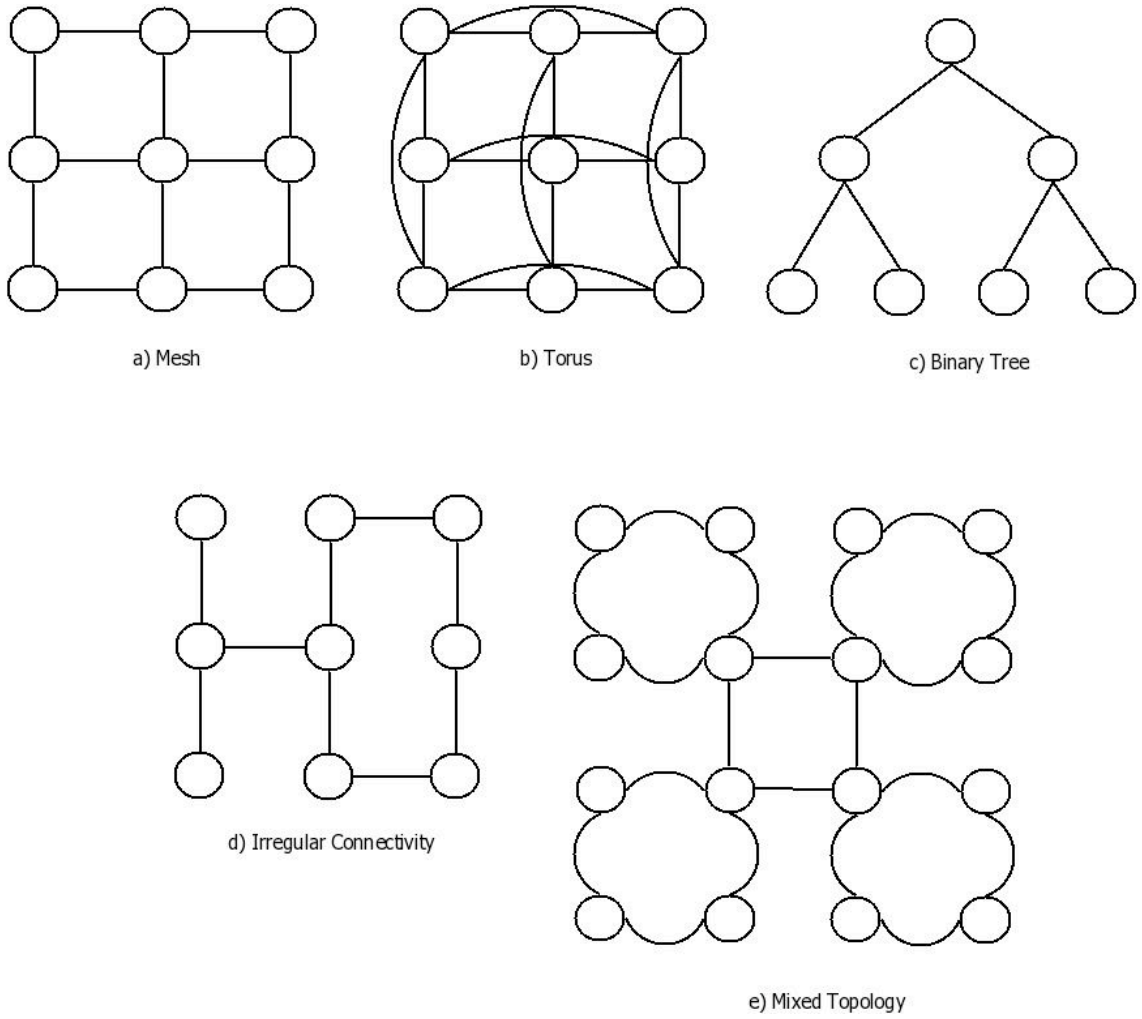

Figure 11: Some commonly found topologies for Switched fabrics

The choice of a topology is very important in a NoC, as it affects several parameters of the SoC such as area, power dissipations and performance; these factors are discussed in detail in Chapter 4 of this report.

EXAMPLES OF NOCS USED IN SOCS

NoCs started off with mesh or torus topologies, because of their simplicity and their ease of integration. A few examples of NoCs from the early days are listed here.

The RAW multiprocessor [13]

This processor consisted of a set of homogeneous tiles comprising of processing elements, storage and routers. The tiles are linked to form as two-dimensional mesh, where the tiles communicate with each other directly.

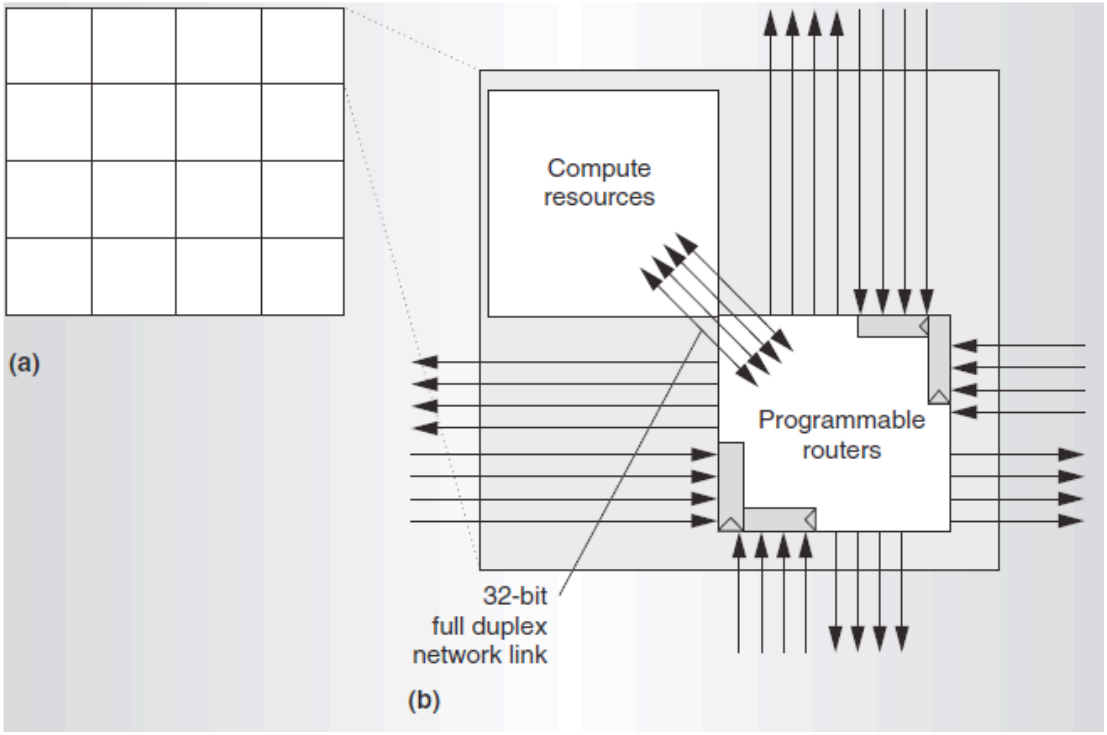

Figure 12: On-chip interconnects in RAW Multiprocessor

The Raw microprocessor comprises 16 tiles (a). Each tile (b) has computational resources and four networks, each with eight point-to-point 32-bit buses to neighbor tiles [13]

The Raw processor has 16 identical, programmable tiles. Each tile contains one static communication router, two dynamic communication routers, an eight-stage, in-order, single-issue, MIPS-style processor, a four-stage, pipelined, floating-point unit, a 32-Kbyte data cache and a 96 Kbytes of software-managed instruction cache. The tiles interconnect using four 32-bit full-duplex on-chip networks, consisting of over 12,500 wires. Two networks are static (routes specified at compile time) and two are dynamic (routes specified at runtime). Each tile only connects to its four neighbors. Every wire is registered at the input to its destination tile. This means that the length of the longest wire in the system is no greater than the length or width of a tile. This property ensures high clock speeds, and the continued scalability of the architecture.

The Nostrum project [14] Used mesh topology to implement the on-chip network communication.

Proteo Network on Chip

The Proteo network on a chip [15] is based on the Torus-based configurations, the k-array 1-cube (one-dimensional torus or ring). The costs of area and power dissipation – proportional to the number of SCRs, Links and average distance – of the ring grew linearly with the number of IP cores. Performance decreased as the size of the NoC grew because the bandwidth is very limited in this topology. The area of the tori is roughly the same as for mesh, where as the power dissipation and performance are better because the average distance is smaller than in mesh.

Octagon – ST Microelectronics

Octagon – ST Microelectronics is a NoC using point-to-point topology. This NoC was designed for network processors. In this network, an octagonal ring connects eight

processors. Recently the Octagon NoC has been extended to a larger NoC named as Spidergon, because of its spider like connections using point-to-point topology.

Core Connect IBM PowerPC processor bus Architecture – IBM

CoreConnect IBM PowerPC processor bus Architecture – IBM [16]

The IBM CoreConnect architecture provides three buses for interconnecting cores, library macros, and custom logic:

- Processor Local Bus (PLB)
- On-Chip Peripheral Bus (OPB)
- Device Control Register (DCR) Bus

Figure 13, illustrates how the Core Connect architecture can be used to interconnect macros in a PowerPC 440 based SOC. High performance, high bandwidth blocks such as the PowerPC 440 CPU core, PCI-X Bridge and PC133/DDR133 SDRAM Controller reside on the PLB, while the OPB hosts lower data rate peripherals. The daisy-chained DCR bus provides a relatively low-speed data path for passing configuration and status information between the PowerPC 440 CPU core and other on-chip macros.

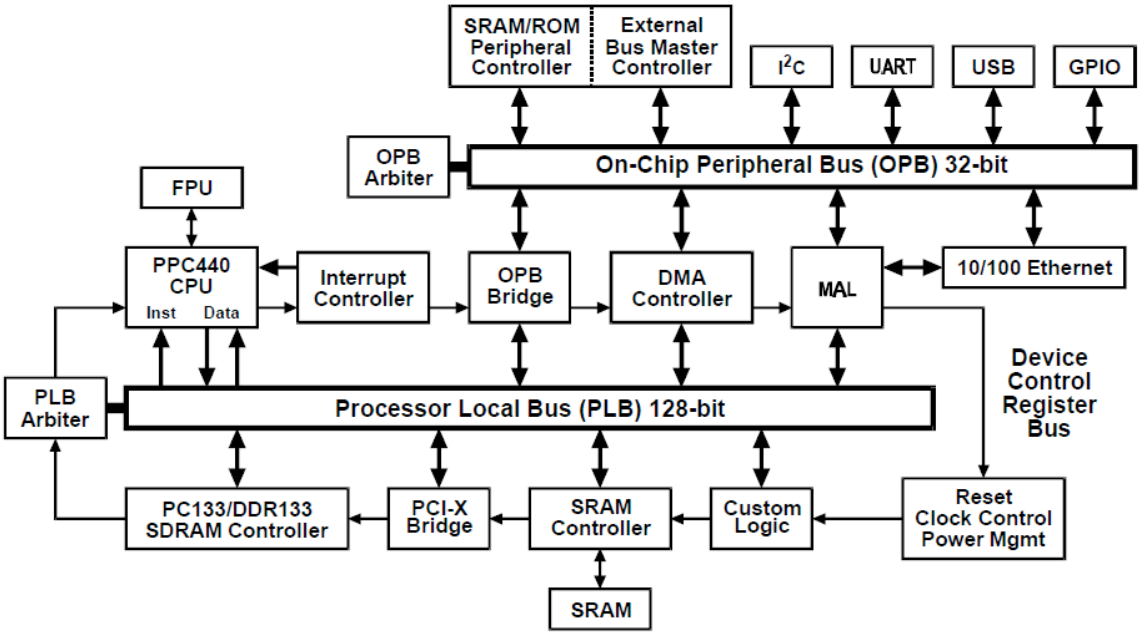

Figure 13: CoreConnect IBM Power PC processor bus

STBus Architecture: ST Microelectronics

The STBus is the result of the evolution of the interconnect subsystem developed for microcontrollers dedicated to consumer applications such as, set top boxes, ATM networks, digital still cameras and others. Such interconnect was born from the accumulation of ideas converging from different sources, such as the transporter (ST20), the Chameleon program (ST40, ST50), MPEG video processing and VCI (virtual component interface) organization.

OMAP Processors: Texas Instruments

OMAP Processor interconnect is a distributed SCR interconnect [6, 7] as shown in the following figure.

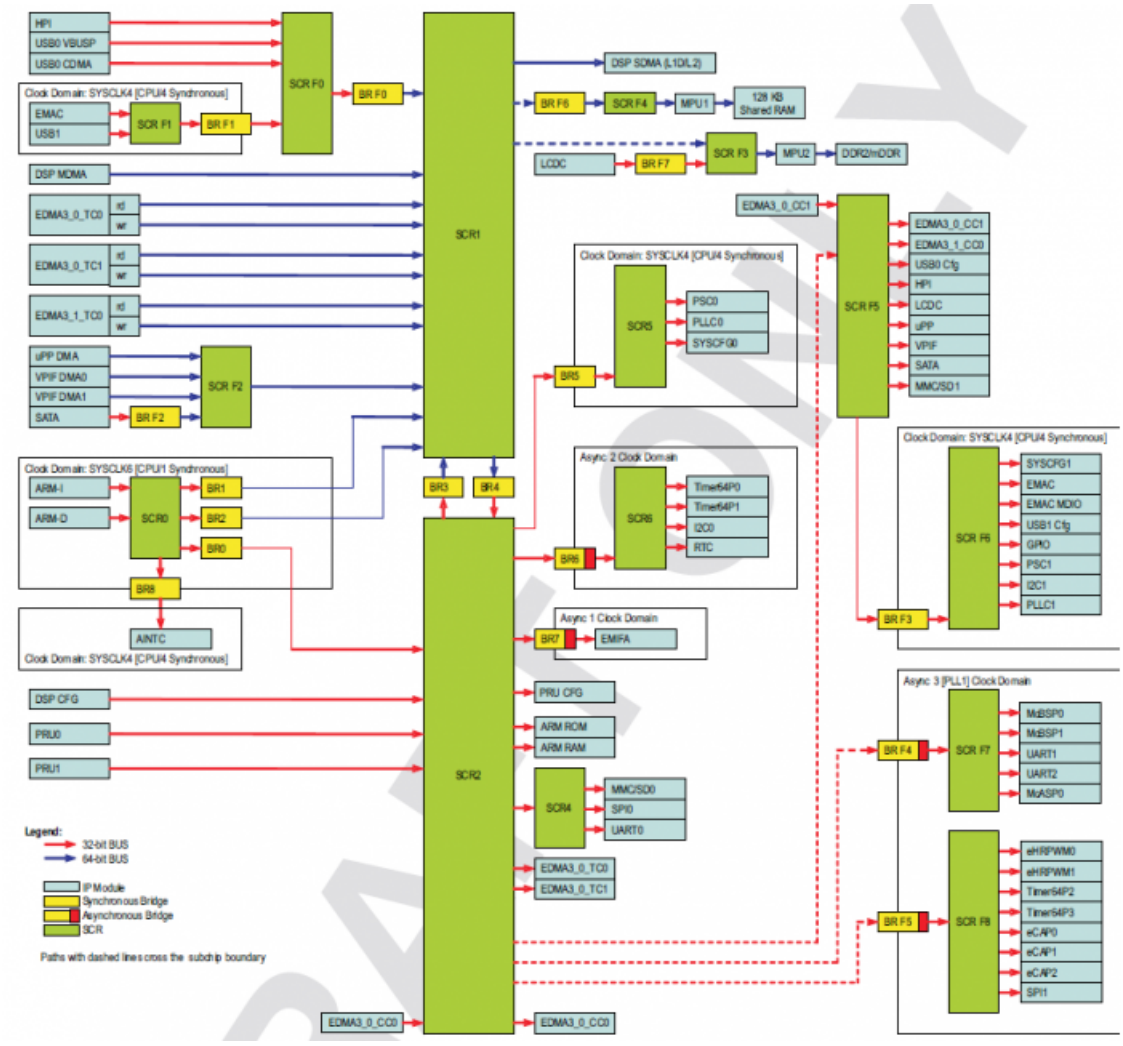

Figure 14: OMAP system interconnect diagram

Chapter 3: Overview of the NoC-Case Study

This chapter introduces the various terms used to discuss the NoC-Case Study.

Texas Instruments developed the Common Bus Architecture (CBA)[1] as a convenient and scalable method of connecting the blocks in a system-on-chip (SoC) without changing the modules themselves.

CBA provides up to 16,000 M-bytes/s (MBps) per interface @ 500 MHz (using a 256-bit data bus) and supports data paths with 8-, 16-, 32-, 64-, 128- or 256-bit data, depending on a trade-off between power, area, and performance.

The CBA also:

- Allows both pipelined burst read and write operations and non-pipelined transfers
- Provides continuous back-to-back transactions, multiple masters, and background master arbitration
- Supports slave-inserted (but not master-inserted) wait states

PROTOCOLS

CBA comprises two different protocols that are referred to as VBUSP and VBUSM. The unit of data and the core control handshaking used to transfer this data is shared by both protocols. This commonalty allows the optimal protocol to be used at various places in the system while maintaining a high degree of compatibility and makes bridging between the protocols a straightforward operation.

Master Slave

In CBA, all data transfers occur between exactly two parties on the interface referred to as the master and the slave.

The master drives all transaction level control information. It also drives the write data and accepts the read data, read status, and write status information.

The slave responds to transactions. For all transactions, this involves accepting and acting on the transaction information. For write transactions, this involves accepting the write data and metering the transfer. For read transactions, this involves driving the read data and metering the transfer

VBUSP Protocol

VBUSP [1] is a very simple and easy to implement protocol that is pended such that only a single transaction can be outstanding at any given time. VBUSP protocol is classified as a point-to-point, pended interface protocol.

Transactions are passed between a master and a slave. They are made up of a collection of data phases that are related to one another by address and order and are transferred as a unit across the interface. At the lowest level, all transactions on VBUSP are either reads or writes as specified by the direction signal.

A write status bus is an optional interface for VBUSP peripherals. The intent of the bus is to signal back to the requesting master when write data has truly landed, as opposed to when write data has left the master ' s boundary.

The master is responsible for specifying all of the parameters surrounding the transaction including the direction, starting address, address progression, and transfer length. The slave is responsible for sourcing the read data and for responding to the transaction by asserting the appropriate ready signal for reads or for writes.

Each data phase of a VBUSP transaction regardless of if its direction, is required to complete on the interface before the next data phase can be presented. This also necessitates that transactions are completed on VBUSP entirely before the next transaction is presented on the interface.

The relatively good performance and simplicity of the VBUSP protocol makes it highly suitable for use in all but the most demanding areas in a SoC.

VBUSM Protocol

VBUSM [1] allows for extensive amounts of transaction pipelining by allowing multiple transactions to be outstanding (like the AXI protocol); this can dramatically increase system performance at a cost of higher complexity and more logic.

The VBUSM protocol represents a significant leap in scalability of performance over VBUSP while maintaining transaction level equivalence. VBUSM provides increased performance capabilities by eliminating the blocking behavior between transactions that is inherent in VBUSP. It allows transaction control signals to be pipelined independently from the data.

A write status bus is a required interface for VBUSM peripherals. The intent of the bus is to signal back to the requesting master when write data has truly landed, as opposed to when write data has left the master ' s boundary (data could be buffered in intermediate bridges).

The master is responsible for driving all of the command interface signals except the ready signal. The slave is given the capability to meter completion of data phases for both reads and writes using a combination of ready signals. It also has direct control over when the read and write status transfers are initiated.

It is restricted for use as a high-speed system bus between CPUs and memory interfaces primarily based on the performance features and cost associated with the VBUSM.

SWITCH CENTRAL RESOURCES (SCR)

The SCR is an N: M crossbar that allows N masters to connect to M slaves. It adds no latency and allows seamless arbitration between the masters and slaves (i.e., no dead cycles inserted by the fabric). The SCR is auto-generated and can restrict access to each slave from any number of the masters.

SCR only supports single protocol (VBUSP or VBUSM), frequency, and bus width. Example SCRs are shown in Table 1. In this example, SCR a crossbar is connecting nine input ports to six output ports, has VBUSM protocol, runs at cpu_clk clock domain, and supports 128-bits bus width. Whereas, SCR B is a VBUSP crossbar connecting four input ports to seven output ports, runs at 1/3 the cpu_clk clock domain, and supports 32-bits bus width.

SCR	Type	Clock Domain	Width	Ports (N: M)
A	VBUSM	cpu_clk	128	9:6
B	VBUSP	cpu_clk/3	32	4:7

Table 1: SCR Examples

Any conversion required for one of the three SCR parameters (protocol type, frequency, or bus width) is handled in bridges.

VBUSM and VBUSP SCRs allow for concurrent data traffic between any of the end-points.

As mentioned previously, the VBUSM protocol allows for pipelining of commands to the various end-points. As such, each of the masters can have multiple commands outstanding; however, VBUSP does not allow for pipelining of commands to the various end-points. Each of the masters can have one command outstanding, at most.

Regardless of the SCR protocol, each master-slave connection is completely independent. So, various masters can have commands outstanding and data in-flight at the same time. When communicating with the same slave end-point, the slave is responsible for arbitrating between the requestors based on priority.

Both VBUSP and VBUSM SCRs transport transactions between a scalable number of master and slave interfaces. They have a selectable number of pipeline stages from 0 to 3 allowing zero clock cycle latency.

Pipelining

When pipelining is enabled, pipeline stages are inserted to break timing paths between the source and destination. These stages can either be before the master decoder, between the decoders and the arbiters, or after the slave arbiter. While the pipeline stages add registers to ease timing, they only insert a single cycle of latency on writes and two cycles of latency on reads (one forward and one reverse) and continue to achieve unity bandwidth.

1. No Pipeline

The following figure represents the main architectural components of a non-pipelined SCR.

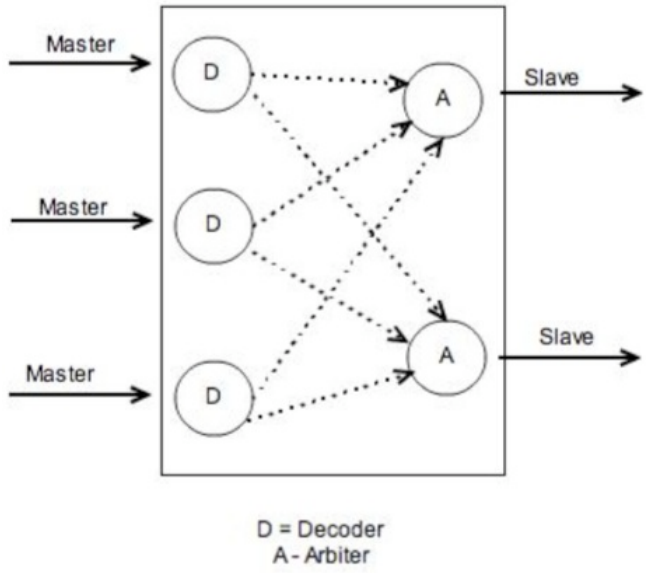

Figure 15: SCR-No pipelines

The Decoder decodes the request from the master, to determine the destination slave. The Arbiter arbitrates between all the requests received by the slave, to determine which request gets serviced.

2. Single Pipeline

The following figure represents the main architectural components of a single pipelined SCR.

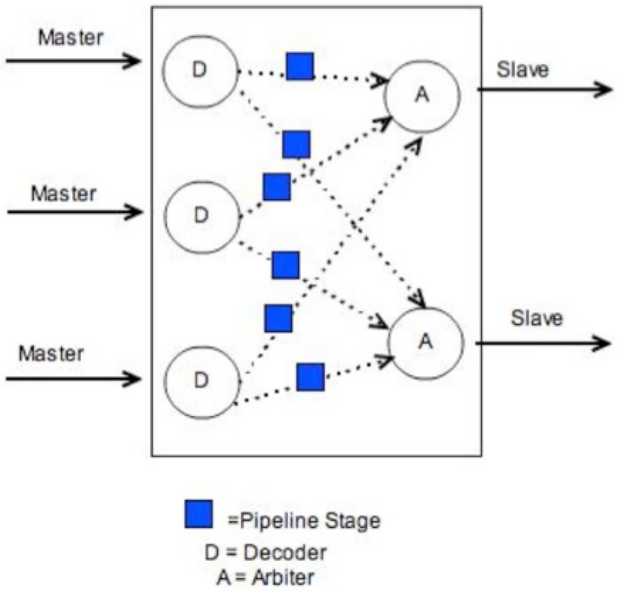

Figure 16: SCR-Single pipelines

In case of a single pipelined SCR, the pipeline is introduced between the Decoders and the Arbiters.

3. Two Pipelines

The following figure represents the main architectural components of a SCR with 2 pipeline stages

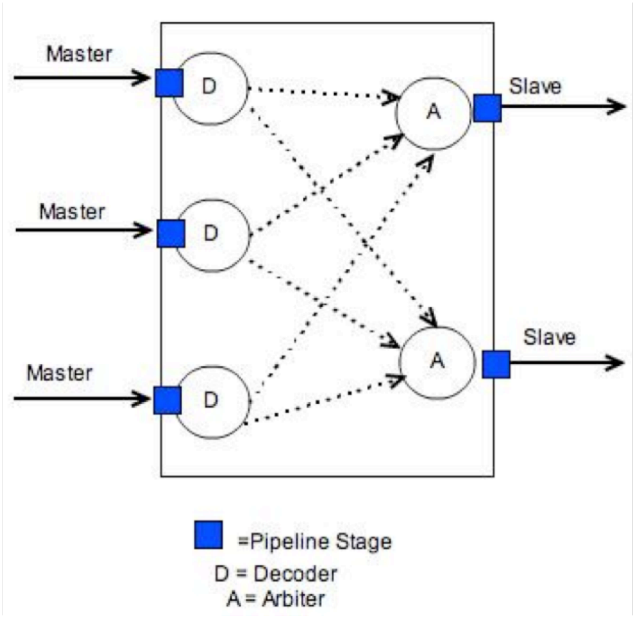

Figure 17: SCR-Two pipelines

In this case, the 2 pipelines are before the decoders and after the arbiters.

4. Three Pipelines

The following figure represents the main architectural components of a SCR with 3 pipeline stages

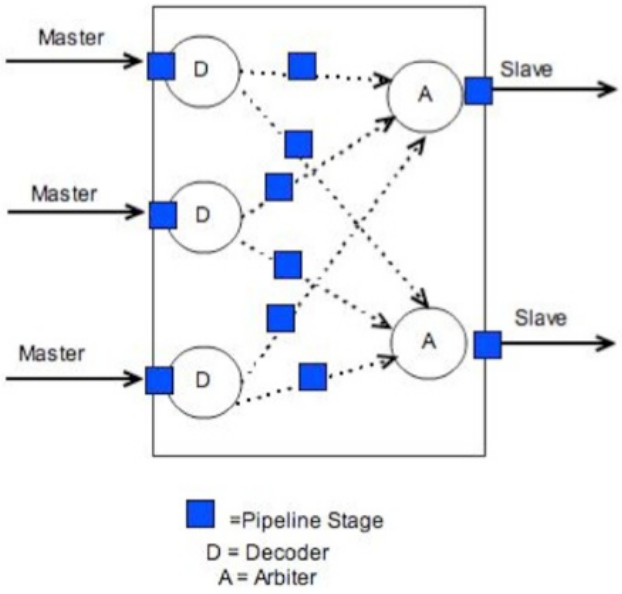

Figure 18: SCR-Two pipelines

In this case, the 3 pipelines are before the decoders, after the arbiters and in-between them.

Arbitration Schemes

Arbitration is used to determine which master interface and will be granted access to the bus in the presence of multiple possible requests.

Bus arbitration usually falls into the following categories

a. Daisy chain arbitration – the bus grant line runs through the connected devices from highest priority to lowest priority with priorities determined by position on the bus. This scheme is simple, but a low priority device may be locked out indefinitely, and the use of a daisy chain grant limits the bus speed.
b. Centralized parallel arbitration – A centralized arbiter chooses from among the devices requesting bus access using multiple request lines and notifies the selected device that it is now bus master. This scheme requires a central arbiter, which may become a bottleneck for bus usage. (The PCI backplane bus uses a central arbitration scheme.)
c. Distributed arbitration by self-selection – These schemes uses multiple request lines, but the device requesting the bus will determine who will be granted bus access. Each device places an identifying code on he request lines and determines the highest priority user that has made a request. (The NuBus uses this scheme.)

Several algorithms for arbitration exist, but we will be examining only a few of them.

Fixed Priority

a. A statistically determined weighted order is used to handle the requests.
b. An ordered list of masters is created.
c. If more than one masters issue a request at a time, the master who occurs first in the list is serviced before the one, which occurs later on in the list.

Round Robin

a. Requests are handled in a cyclic manner. All the masters are checked in a cyclic pattern for requests.
b. If a request exists while the issuing master is being checked, it is serviced.

Round Robin Fixed Priority

a. This is a 2 tier approach, combining the round robin and fixed priority
b. Requests in each level are handled through the round robin arbitration scheme, and the winners of all the levels are handled through the fixed priority among levels
c. In cases where the priority is determined dynamically, by analyzing the traffic occurring in the chip, the scheme is called Dynamic Round Robin Fixed Priority.

BRIDGES

Bridges are major components in CBA based systems. Like SCRs, bridges are considered part of the infrastructure and are not themselves considered initiators or targets. Bridges are not required in all systems but are typically used to:

1. Connect bus segments running at different frequencies
2. Connect bus segments of different data path widths
3. Connect bus segments with different protocols (VBUSP vs. VBUSM)

Bridges are also designed in a way to increase overall throughput. They can be used in a variety of places to connect initiators, targets, central resources, or to connect one bridge to another. Table 2 shows the example bridges.

Bridge	Type (Mstr: Slv)	Width (Mstr: Slv)	Clk (Mstr: Slv)	Read FIFO Ctrl	Read FIFO data	Read FIFO burst size	Write FIFO data	Write FIFO burst size	Cod FIFO Depth	Status FIFO Depth
1	P: M	128:128	1:1	N/A	N/A	N/A	N/A	N/A	N/A	N/A
2	M-M	64:128	1:1	8	8	1	3	2	3	2
3	M-M	32:64	1:2	3	7	1	6	4	2	2
4	M-P	64:32	2:1	2	2	1	3	0	2	2

Table 2: Bridge Examples

- Bridge type: This is the protocol conversion provided by the bridge. In Table 2, an “M” indicates VBUSM and a “P” indicates VBUSP. The first letter marks the protocol of the bridge master and the second letter marks the protocol of the bridge slave. For example, bridge #4 connects a master VBUSM bus to a slave VBUSP bus.
- Width: This is the bus width conversion in bits. The first number is the bus width of the bridge master and the second number is the bus width of the bridge slave. For example, bridge #3 has a 32-bit wide master bus and a 64-bit wide slave bus.
- Clock: This is the relative clock ratio of the master clock to the slave clock. For example, bridge #4 could be a bridge between a module at one-third the

CPU rate and a module at one-sixth the CPU rate, so the clock bridge parameter is 2:1. Any clock scaling must be global, unless an asynchronous bridge is used between components.

- Read first-in-first-out (FIFO) control: This parameter indicates the number of outstanding read commands that can be present in the bridge. In VBUSP, only a single transaction can be outstanding at any given time; the read FIFO control parameter is only used in bridges where the master has VBUSM protocol.
- Read FIFO data: This parameter indicates the FIFO depth used for read response data. FIFO entries are indicated in terms of the widest port of the bridge (i.e., read FIFO data = 8 for bridge #2 means the FIFO depth is 8 * 128-bit). This parameter is only used in bridges where the master has VBUSM protocol.
- Read FIFO burst size: This parameter controls the bursting attributes of the read response back to the master. The burst size, as with the FIFO size, is in terms of data phases with respect to the wider of the two bridge data buses. The read FIFO burst size parameter is only used in bridges where the master has VBUSM protocol. Data is buffered in the bridge until the read burst size is reached. Time-out mechanism exists to avoid indefinite waiting for small data amounts. A value of 0 for read FIFO burst size means there is no wait.
- Write FIFO data: This parameter is comparable to that for reads. It is the FIFO depth for writes data. The write FIFO data parameter is only used in bridges where the master has VBUSM protocol.
- Write FIFO burst size: This parameter is comparable to that for read FIFO burst size. It indicates the write bursting attributes for the bridge. It is only

used in bridges where the master has VBUSM protocol. Data is buffered in the bridge until the write burst size is reached. Time-out mechanism exists to avoid indefinite waiting for small data amounts. A value of 0 for write FIFO burst size (as in bridge #4) means there is no wait.

- Command FIFO depth: This parameter indicates the total number of commands that can be accepted by the bridge at any point in time. It is used only in bridges where the master has VBUSM protocol.
- Status FIFO depth: This indicates the depth of the FIFO used for write status back to the master. It is used only in bridges where the master has VBUSM protocol.

Chapter 4: Analysis of the architectural modifications to SCRs

As discussed in the previous chapters, the main core of a NoC is the Switched Fabric. The area cost, power consumption and performance of a NoC, depends directly on the topology of the Switched Fabric. The major contributors to this are the Switching Central Resources (SCRs) and the Links (as shown in Figure 19). Any architectural modifications to a SCR, directly impacts the NoC's metrics.

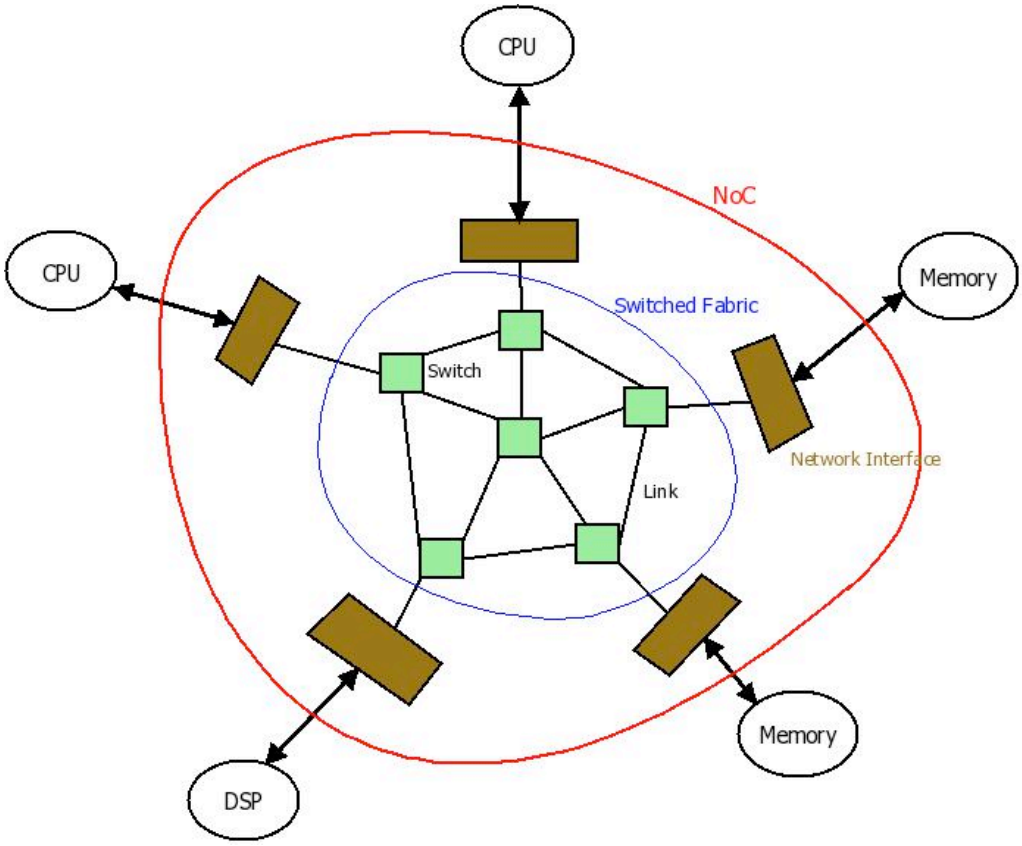

Figure 19: Network on a Chip (NoC), Overview

This chapter presents the implications of architectural design choices on the performance and cost metrics namely latency, area, and power of a SCR.

The silicon area of a NoC depends directly on the topology of the Switched Fabric. The area of the SCRs and the Links (Chapter 3) can be computed by gate level synthesis. A first order approximation of the area consumed by the Links can be obtained by, the number of global wires (links * width of respective Link).

The power consumption cost in a NoC depends on the number of NoC components, that are active due to the data in the NoC itself, and those, which are independent of the data such as the static power and the clock power.

The performance of a NoC depends on various factors, which includes the topology of the Switched Fabric. For a high performance topology (such as, large bandwidth) tend to have a high area cost. A trade-off between area and performance is an important criterion for the design.

The critical path length of a component is defined by the maximum delay value among the delays for signals to propagate within the given component. A smaller critical path length implies, the component can be operated at a higher frequency, increasing its performance. The critical path slack is defined as the difference between clock period and the critical path length. A negative value of critical path slack implies that the component has not met the performance target.

Switching Central Resources (SCR) mainly consists of a switch matrix, network interfaces with a communication protocol and routing and arbitration techniques (as discussed in the previous chapters). The main architectural and design parameters of the SCRs include [1, 9, 10, 11]

1. Number of Masters and Slaves (number cores initiating requests and number of cores servicing them)

2. The switching interconnects (Switching Matrix) topologies such as Crossbar, Bus, Ring, Star, and Multi-stage Interconnect Network (MIN). Currently in SoC, Crossbar and MIN-static are dominant topologies, as the IP cores on chip are limited to a maximum of 16 to 64. As these IP cores increase, a greater variety of network implementations are expected.
3. The standard bus protocol to interface with, such as AXI (from ARM), VBUSP and VBUSM (from Texas Instruments), OCP (from OCP-IP)
4. Bus Width
5. The number of pipeline stages
6. The Voltage Domain
7. The frequency of implementation
8. The arbitration scheme such as fixed priority, first come first serve, round robin, hybrid

The choice of these architectural and design parameters have a large impact on the performance, power dissipation and area of the Switched Fabric and hence the NoC. With the increase in the number of cores communicating, the sizes of the SCRs are increasing. Previous techniques [30, 31, 32], with similar analyses on NoCs do provide us a good reference. These techniques involve complex calculations, where as the analyses in this report provide a higher level of abstraction for quick estimations, which are experimentally proven.

EXPERIMENTAL SETUP

The analysis is carried out using a Common Bus Architecture (developed by Texas Instruments), which uses a cross bar switching matrix and VBUSP and VBUSM protocols (similar to AXI by ARM) to communicate. As the frequency of

implementation, bus width and the bus protocol are a part of the specifications of any design, the variations with respect to number of masters, number of slaves, arbitration schemes (round robin, fixed priority and dynamic round robin fixed priority), voltage domain, number of pipeline stages are analyzed. These experimental analyses provides a means of high level estimations and analysis of SCR in the early stages of the design process, which helps avoid re-design in the middle of the design cycle which leads to an increase in the time to market. Since these parameters are currently most widely used (switch network of crossbars [2] and AXI similar multiple address issue bus protocols [3, 4]) in SoCs, these analysis results would be useful estimates for many similar SCRs.

The data in this report is normalized, in order to honor a Non Disclosure Agreement with Texas Instruments. The experimental results are obtained though Design Compiler (A synthesis tool by synopsis) and in-house power estimation tools, using a 45nm library class similar to TSMC.

Area and latency are estimated through Design compiler. The significant parameters used are as follows:

- Ideal networks for Master clock and reset input ports
- 20% clock transitions
- 50% of clock period as input and output delay of the ports
- Synthesized to minimize area
- Maximum of 8 metal layers with 70% utilizations
- Hierarchy in design is flattened
- Voltage: 0.9V and 1.0V
- Frequency 250MHz and 500MHz
- Temperature -40ºC, to create worst case silicon scenario

Power is estimated using Texas Instruments in-house tools. The significant parameters used are as follows:

- Activity factors based on application scenarios
- Number of metal layers used 8
- Voltage: 0.9V and 1.0V
- Frequency 250MHz and 500MHz
- Using high performance standard cells from the 45nm library

ANALYSIS WITH INCREASING THE NUMBER OF SLAVES

The effects on the critical path length, area and power are analyzed with variations in the number of slaves.

Specification/Parameters

The parameters used for this analysis are as follows:

Protocol = VBUSM

Number of Masters = 4

Number of Slaves = Variable

Frequency of Synthesis = 250MHz

Voltage Domain = 1V

Temperature = -40ºC

Number of pipeline stages = 2

Data path width = 128 bits

Arbitration = Round Robin Fixed Priority

Effects on the Critical Path Length

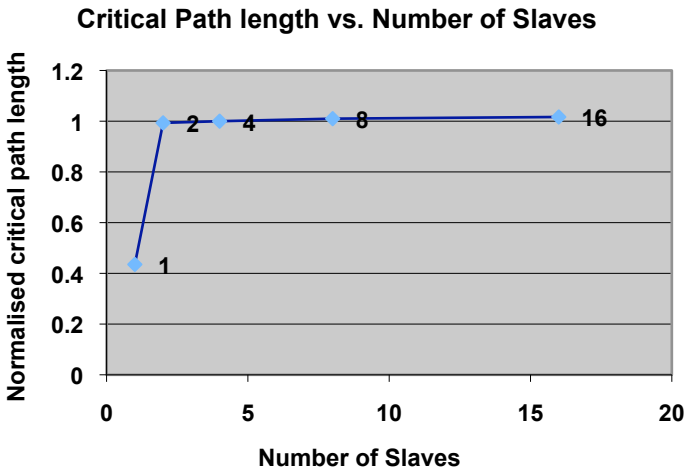

Figure 20: Effect on critical path length due to variation in number of Slaves

Number of Slaves	1	2	4	8	16
Normalized Critical Path Length	0.435	0.993	1	1.009	1.016

Table 3: Effect on critical path length due to variation in number of Slaves

Conclusion/Analysis

As the number of slaves increases the Critical path slack increases by a very small amount. The critical path length almost doubles when the number of slaves is increased from 1 to 2. This is due to the fact that when there is more than one slave, there is a decoder required for each master. In cases where there are multiple slaves, by increasing the number of slaves, the complexity of the decoders' increases, while that of the arbiters remains the same. In this range, the critical path length increases linearly by about 1% with the doubling of the number of slaves.

Effects on the Area (Gate count)

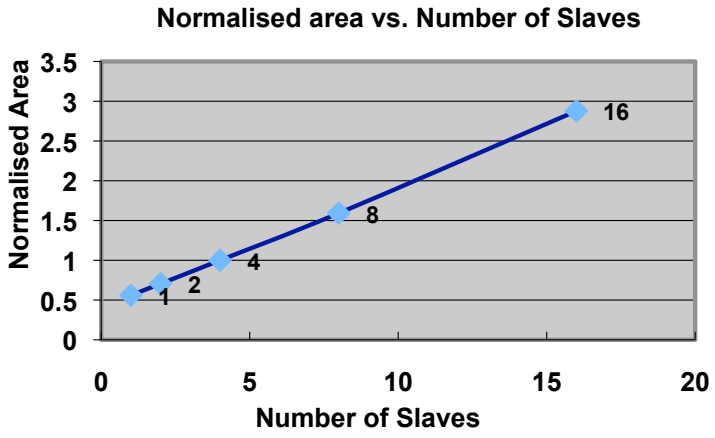

Figure 21: Effect on area due to variation in number of Slaves

Number of Slaves	1	2	4	8	16
Normalized Area	0.560	0.705	1	1.592	2.877

Table 4: Effect on area due to variation in number of Slaves

Conclusion/Analysis

The area increases linearly with the increase in number of slaves. Addition of a slave involves an addition of another arbiter as well as increasing the complexity of the decoder. In this experiment, doubling the number of slaves results in an increase in area by about 50%.

Effects on the Total Power

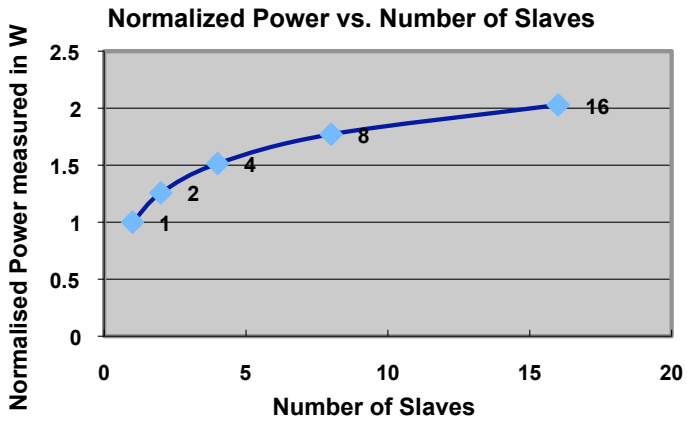

Figure 22: Effect on total power due to variation in number of Slaves

Number of Slaves	1	2	4	8	16
Normalized Power	1	1.257	1.514	1.771	2.028

Table 5: Effect on total power due to variation in number of Slaves

Conclusion/Analysis

Even though the plot looks exponential when all points are considered; however the first point has to be ignored, because with only one slave, there are no decoders, making it unique among the other points. Hence, considering the valid points, the plot is linear with a step size of 25%. The effects on the critical path length, area and power are analyzed with variations in the number of masters.

ANALYSIS WITH INCREASING THE NUMBER OF MASTERS

Specification/Parameters

The parameters used for this analysis are as follows:

Protocol = VBUSM Temperature = -40°C

Number of Masters = Variable Number of pipeline stages = 2

Number of Slaves = 4 Data path width = 128 bits

Frequency of Synthesis = 250MHz Arbitration = Round Robin

Voltage Domain = 1V

Effects on the Critical Path Length

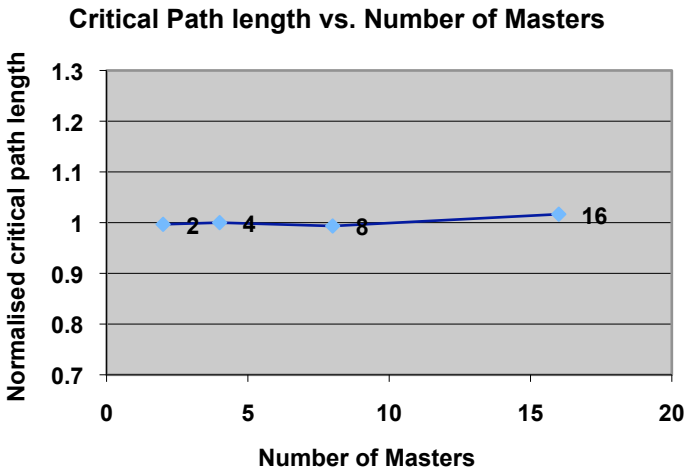

Figure 23: Effect on critical path length due to variation in number of Masters

Number of Masters	2	4	8	16
Normalized Critical Path Length	0.996	1	0.993	1.016

Table 6: Effect on critical path length due to variation in number of Masters

Conclusion/Analysis

As the number of masters increases, the critical path slack increases by a very small amount. With the increase in masters, the complexity of the arbiter in the critical path increases, and the number of decoders increase. However, the increase in number of decoders does not add to the critical path (Figure 12).

Effects on the Area (Gate count)

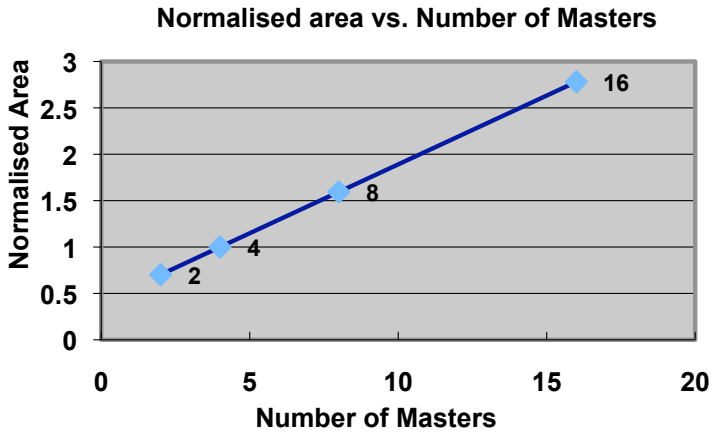

Figure 24: Effect on area due to variation in number of Masters

Number of Masters	2	4	8	16
Normalized Area	0.701	1	1.594	2.780

Table 7: Effect on area due to variation in number of Masters

Conclusion/Analysis

As the number of masters' increase, the area increases linearly. Each addition of a master, adds a decoder and a little increase in complexity of arbiter. The equation of the above line is Area = 0.15(Number of Masters) + 0.4

Effects on the Total Power

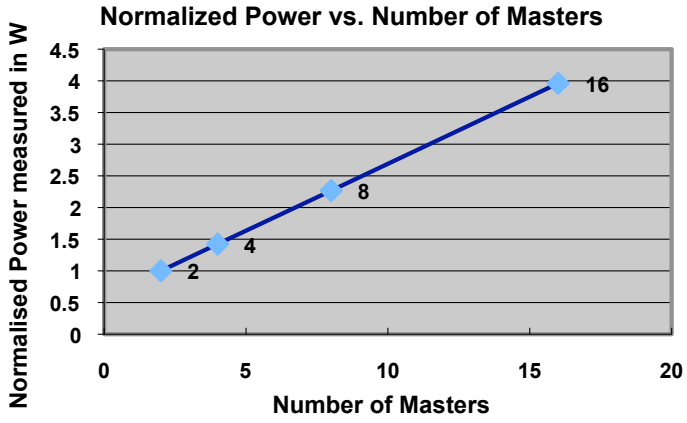

Figure 25: Effect on total power due to variation in number of Masters

Number of Masters	2	4	8	16
Normalized Power	1	1.424	2.267	3.961

Table 8: Effect on total power due to variation in number of Masters

Conclusion/Analysis

As the number of masters' increase, the power increases linearly. Each addition of a master, adds a decoder and a little increase in complexity of arbiter. The equation of the above line is approximately Power = 0.2(Number of Masters) + 0.6

ANALYSIS WITH CHANGES IN ARBITRATION SCHEMES

The effects on the critical path length, area and power are analyzed with variations in the arbitration scheme used.

Specification/Parameters

The parameters used for this analysis are as follows:

Protocol = VBUSM

Number of Masters = Variable

Number of Slaves = Variable

Frequency of Synthesis = 250MHz

Voltage Domain = 1V

Temperature = -40°C

Number of pipeline stages = 2

Data path width = 128 bits

Arbitration = Variable

Effects on the Critical Path Length

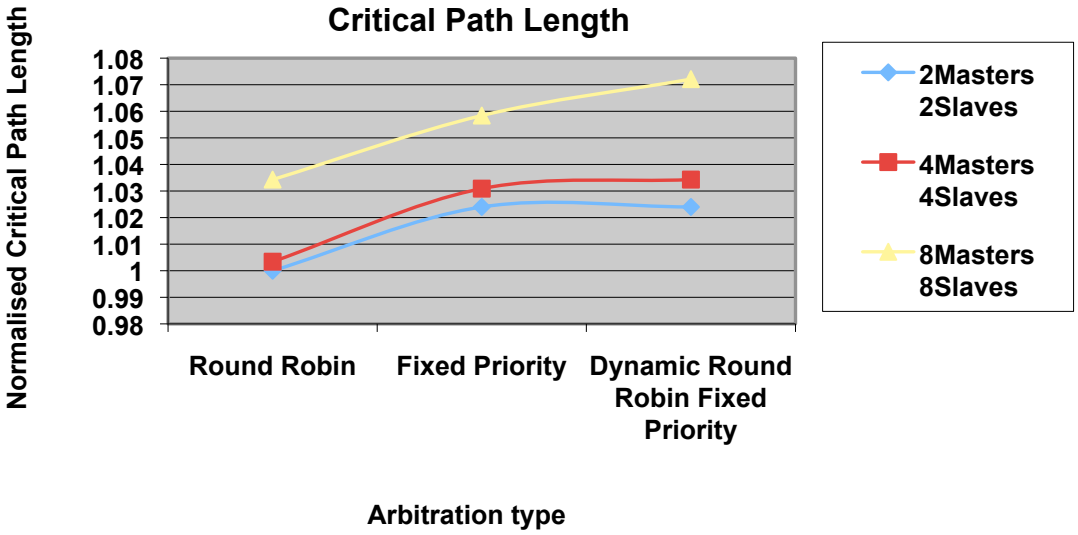

Figure 26: Effect on critical path length due to variation in arbitration technique

Critical Path Length	Round Robin	Fixed Priority	Dynamic Round Robin Fixed Priority
2Masters 2Slaves	1	1.024	1.024
4Masters 4Slaves	1.0034	1.0309	1.0343
8Masters 8Slaves	1.0343	1.0584	1.0721

Table 9: Effect on critical path length due to variation in arbitration technique

Conclusion/Analysis

The critical path length is best for Round Robin (RR) arbitration, followed by Fixed Priority (FP) and then Dynamic Round Robin Fixed Priority (DRRFP). In the case of 2 Masters and 2 Slaves, DRRFP is equivalent to FP as there are only 2 slaves. For any given arbitration scheme, with the increase in masters and slaves, the complexity in the decoder and the arbiter increases, increasing the critical path length. For small SCRs, the difference between in critical path length with DRRFP and FP is very small, but as the size of the SCR increases, the difference in critical path lengths become significant.

Effects on the Area (Gate count)

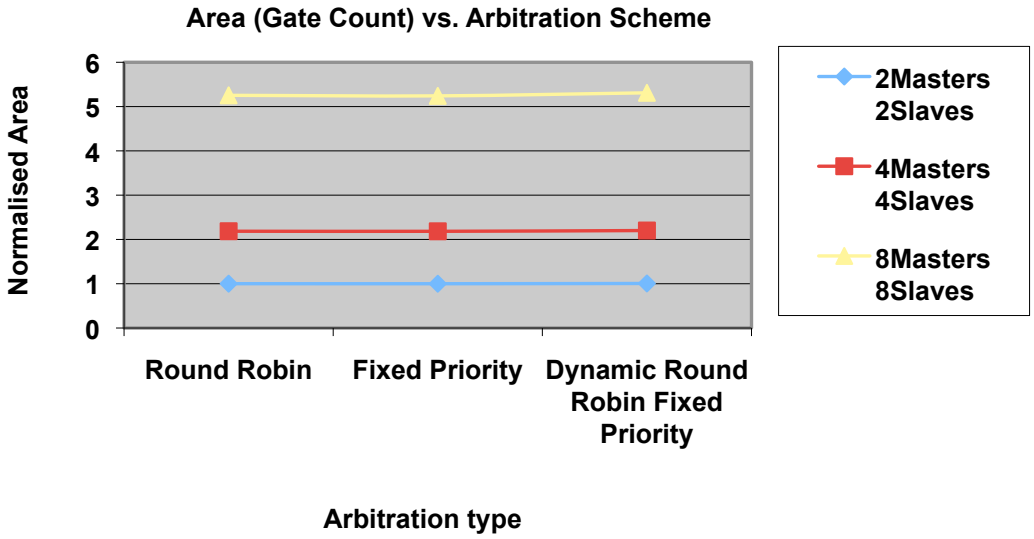

Figure 27: Effect on area due to variation in arbitration technique

Normalized Area	Round Robin	Fixed Priority	Dynamic Round Robin Fixed Priority
2Masters 2Slaves	1.001	1	1.005
4Masters 4Slaves	2.185	2.183	2.201
8Masters 8Slaves	5.254	5.241	5.310

Table 10: Effect on area due to variation in arbitration technique

Conclusion/Analysis

The area is almost constant with a modification only to the arbitration schemes. As seen with a previous analysis, for any particular arbitration scheme, the growth in area with respect to masters and slaves is linear individually, making a quadratic growth when both the masters and slaves are modified.

Effects on the Total Power

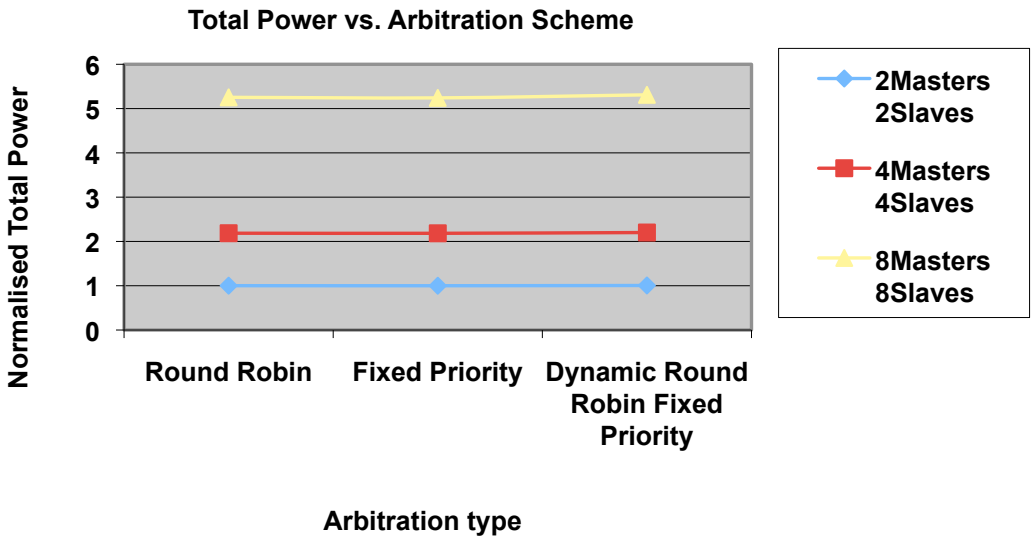

Figure 28: Effect on total power due to variation in arbitration technique

Normalized Power	Round Robin	Fixed Priority	Dynamic Round Robin Fixed Priority
2Masters 2Slaves	1.001	1	1.005
4Masters 4Slaves	2.185	2.183	2.201
8Masters 8Slaves	5.254	5.241	5.310

Table 11: Effect on total power due to variation in arbitration technique

Conclusion/Analysis

The power is almost constant with a modification to the arbitration schemes. As seen with a previous analysis, for any particular arbitration scheme, the growth in power with respect to masters and slaves is linear individually, making a quadratic growth when both the masters and slaves are modified. This behavior is, similar to that seen during area analysis.

ANALYSIS WITH CHANGES IN NUMBER OF PIPELINE STAGES

The effects on the critical path length, area and power are analyzed with variations in the number of pipeline stages in a SCR

Specification/Parameters

The parameters used for this analysis are as follows:

Protocol = VBUSM

Number of Masters = 8

Number of Slaves = 8

Frequency of Synthesis = 250MHz

Voltage Domain = 1V

Temperature = -40°C

Number of pipeline stages = variable

Data path width = 128 bits

Arbitration = Variable

Effects on the Critical Path Length

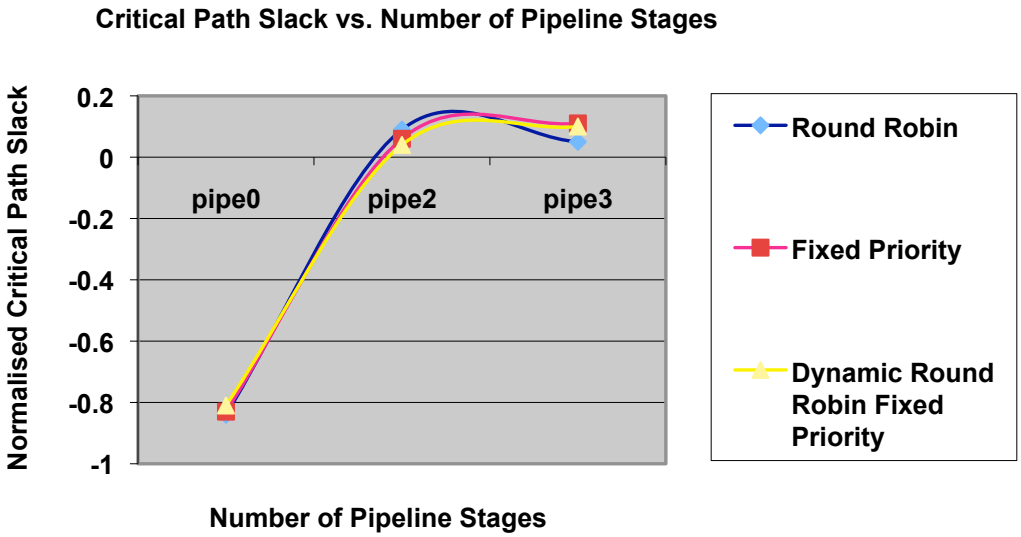

Figure 29: Effect on critical path length due to variation in number of pipeline stages

Critical Path Slack	Pipe 0	Pipe 2	Pipe 3
Round Robin	-0.84	0.09	0.05
Fixed Priority	-0.83	0.06	0.11
Dynamic Round Robin Fixed Priority	-0.81	0.04	0.1

Table 12: Effect on critical path length due to variation in number of pipeline stages

Conclusion/Analysis

At a voltage supply of 1V, it is difficult to timing close the SCRs without any pipeline stages, shown by the negative values of normalized critical slack. At voltage of 1V with two or three pipeline stages, this SCR closes timing.

Effects on the Area (Gate count)

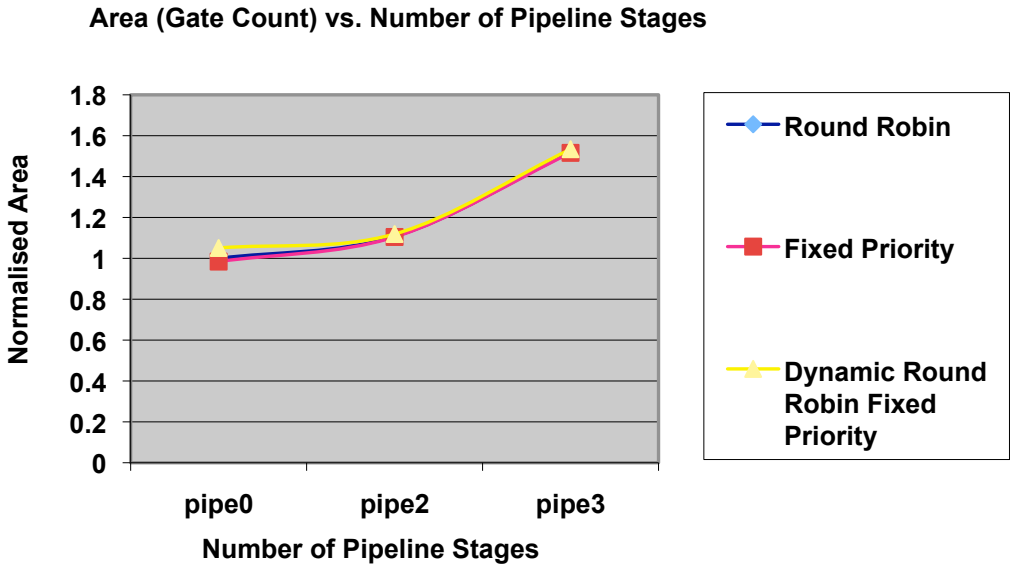

Figure 30: Effect on area due to variation in number of pipeline stages

Normalized Area	Pipe 0	Pipe 2	Pipe 3
Round Robin	1.0	1.107	1.519
Fixed Priority	0.984	1.104	1.516
Dynamic Round Robin Fixed Priority	1.050	1.119	1.533

Table 13: Effect on area due to variation in number of pipeline stages

Conclusion/Analysis

The area increases by about 10% with the addition of 2 pipeline stages and by about 50% with the addition of 3 pipeline stages. If M is the number of masters and S is the number of Slaves, with the addition of 2 pipeline stages, the increase in area is directly proportional to M + S. Where as in the addition of 3 pipeline stages, the increase in area is directly proportional to M+S+MS.

Effects on the Total Power

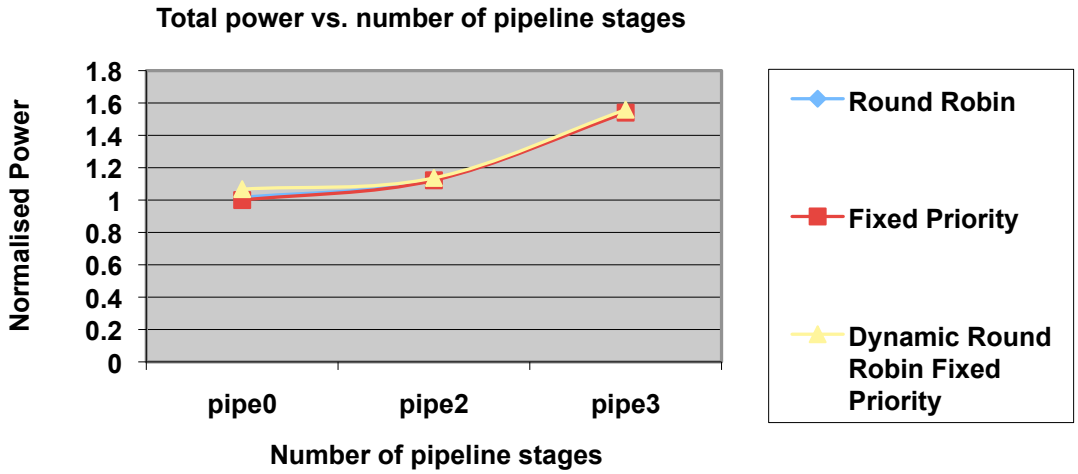

Figure 31: Effect on total power due to variation in number of pipeline stages

Normalized Power	Pipe 0	Pipe 2	Pipe 3
Round Robin	1.016	1.124	1.543
Fixed Priority	1	1.122	1.540
Dynamic Round Robin Fixed Priority	1.067	1.136	1.558

Table 14: Effect on total power due to variation in number of pipeline stages

Conclusion/Analysis

The power is almost constant with a modification to the arbitration schemes. As seen with a previous analysis, for any particular arbitration scheme, the growth in power with respect to masters and slaves is linear individually, making a quadratic growth when both the masters and slaves are modified. The power increases by about 10% with the addition of 2 pipeline stages and by about 50% with the addition of 3 pipeline stages. If M is the number of masters and S is the number of Slaves, with the addition of 2 pipeline stages, the increase in power is directly proportional to M + S. Where as in the addition of 3 pipeline stages, the increase in power is directly proportional to M+S+MS.

ANALYSIS WITH CHANGES IN VOLTAGE SUPPLY

The effects on the critical path length, area and power are analyzed with variations in the voltage supply.

Specification/Parameters

The parameters used for this analysis are as follows:

Protocol = VBUSM

Number of Masters = 8

Number of Slaves = 8

Frequency of Synthesis = 500MHz

Temperature = -40ºC

Number of pipeline stages = variable

Data path width = 128 bits

Voltage Domain =Variable

Arbitration = Dynamic Round Robin Fixed Priority

Effects on the Critical Path Length

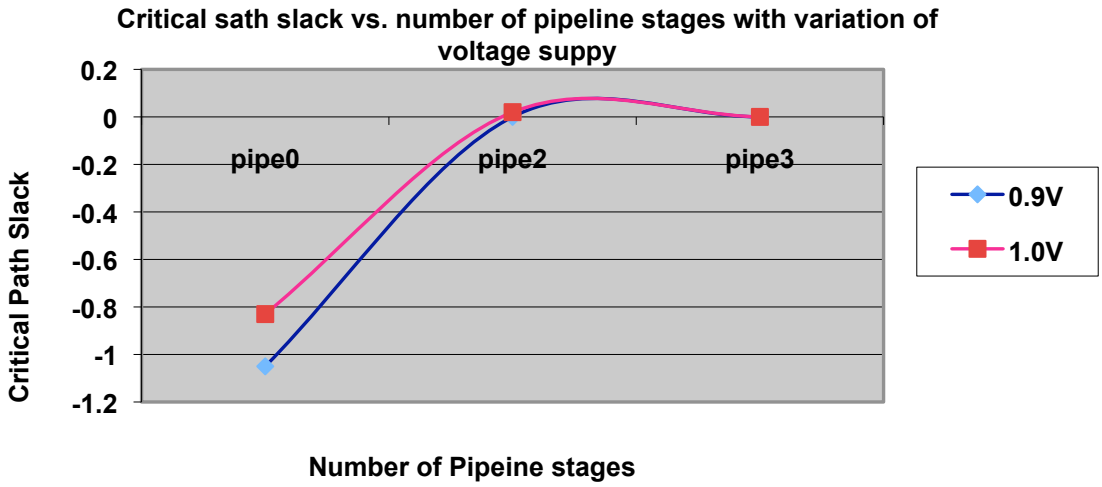

Figure 32: Effect on critical path length due to variation supply voltage

Critical Path Slack	Pipe 0	Pipe 2	Pipe 3
0.9V	-1.05	0	0
1.0V	-0.83	0.02	0

Table 15: Effect on critical path length due to variation supply voltage

Conclusion/Analysis

At 1V, it is difficult to timing close the SCRs without any pipeline stages, shown by the negative values of normalized critical slack. At 1V and two or three pipeline stages, this SCR closes timing.

Effects on the Area (Gate count)

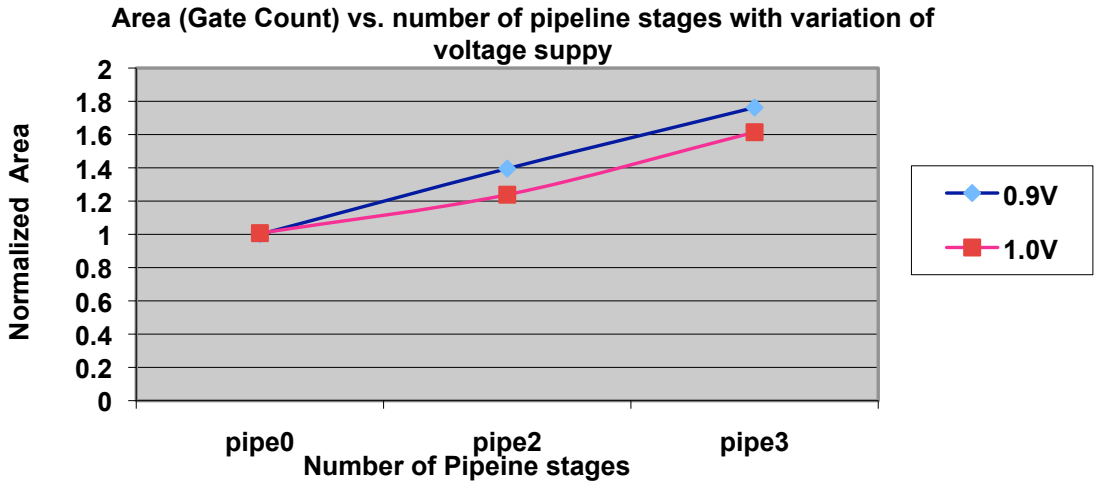

Figure 33: Effect on area due to variation supply voltage

Normalized Area	Pipe 0	Pipe 2	Pipe 3
0.9 V	1.000	1.395	1.762
1.0 V	1.007	1.238	1.614

Table 16: Effect on area due to variation supply voltage

Conclusion/Analysis

A decrease in voltage supply from 1.0V to 0.9V led to an increase in area. The main contributors are increased buffer count and larger drive strength cells usage, to meet timing.

Effects on the Total Power

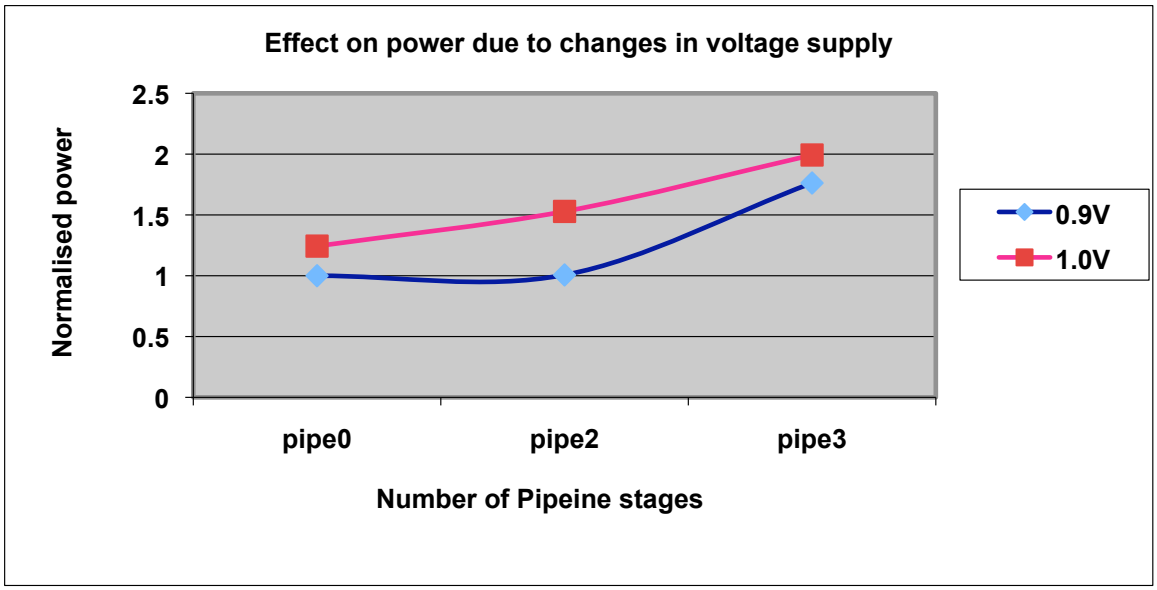

Figure 34: Effect on total power due to variation supply voltage

Normalized Power	Pipe 0	Pipe 2	Pipe 3
0.9 V	1	1.006	1.762
1.0 V	1.243	1.528	1.992

Table 17: Effect on total power due to variation supply voltage

Conclusion/Analysis

As expected, the power consumption, of the module, reduces with a decrease in voltage supply from 1.0V to 0.9V.

SUMMARY OF THE ABOVE EXPERIMENTS TO HELP IN EARLY DESIGN ESTIMATIONS

For an SCR based NoC architecture, we have proven using the above experiments that, we can estimate the area for any architecture change with respect to the number of Masters/Slave combinations to be increasing approximately linearly. With variations in the number of masters and slaves, one expects that the area might behave differently due to the modifications of complexity in the number of decoders and arbiters including their logic cones. These variations can be considered liner and have been proven in the above experiments. This helps designers to make an early safe judgment on the impact on area, and avoid multiple experiments that are generally done in current industry.

When designers use a different technology, initial 2 points of analysis is required. With these points, the results for further architectural modifications can be extrapolated similar to those shown from the above experiments.

In brief, Table 18 contains the high level estimates

Variation Parameters	**Area**	**Power**	**Critical path Length**
Increase in slaves	Linear increase, quadratic variation with increase in masters	Linear	Negligible/small change
Increase in masters	Linear increase, quadratic variation with increase in masters	Linear	Negligible/small change
Variation in arbitration schemes	Almost constant	Almost constant	DRRFP > FP > RR
Increase in pipeline stages	Pipe0 < Pipe2(1.1*Pipe0) < Pipe3(1.5*Pipe0)	Similar to Area behavior	Timing problems seen in Pipe0
Increase in supply voltage	Area is inversely proportional to voltage	Power is proportional to voltage (The increase in voltage supply, dominates decrease in area)	Pipelining helps to ease out timing closure. Lesser effort with higher voltage supply.

Table 18: A short summary of the experiments

Chapter 5: Analysis of the main SCR in a TI Subsystem –Case Study

The interconnect of the Subsystem is shown in Figure 35:

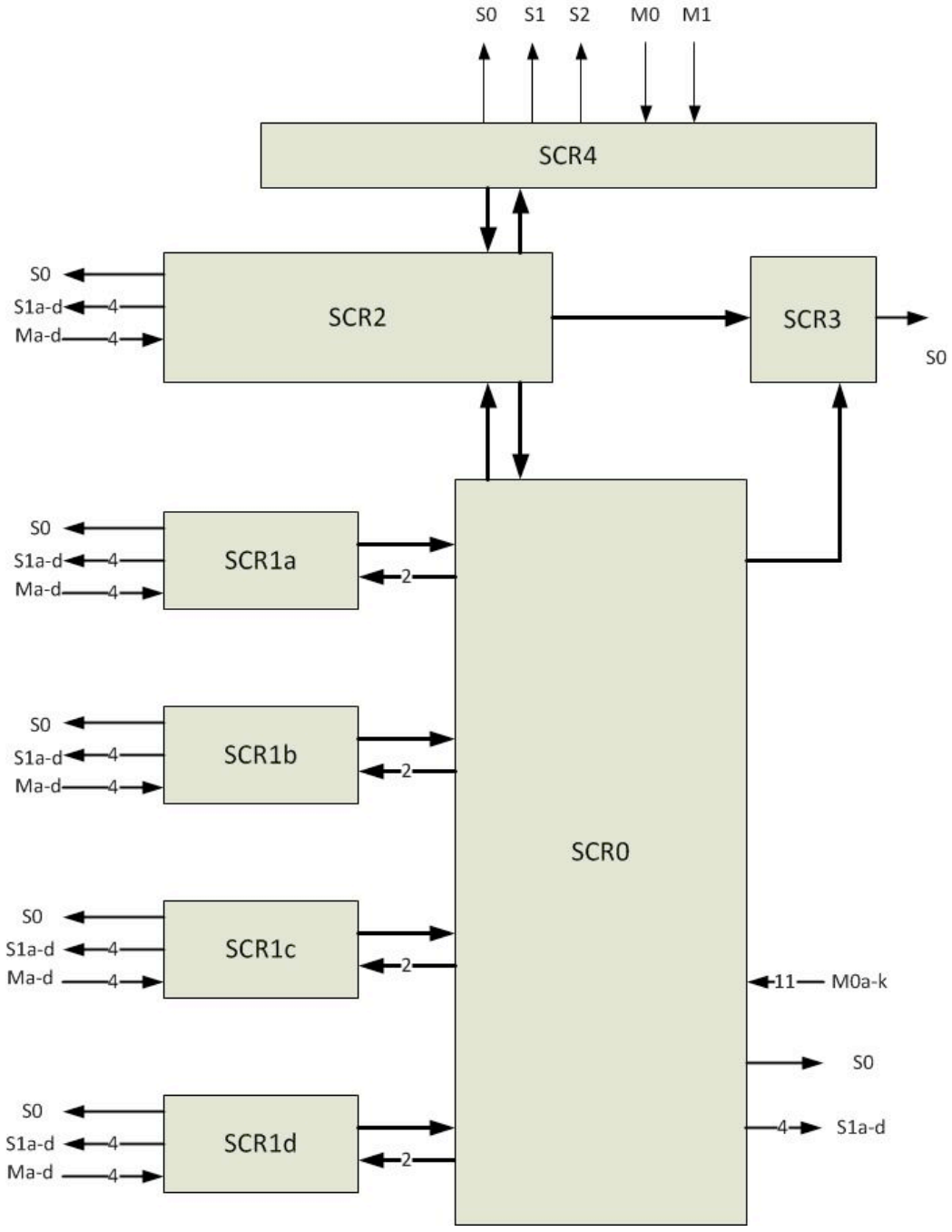

Figure 35: Interconnect architecture of the Texas Instruments Sub-system

The main SCR is the largest SCR among the distributed network of SCRs, which connects and establishes the communication among all the modules in the system. It consists of 16 masters and 15 slaves, communicating through the VBUSM protocol.

The performances indexes are measured and analyzed to obtain the most suitable architectural configuration. The experimental setup is same as described in the previous chapter.

ADDITIONAL TERMINOLOGY

Connectivity in this context of SCR refers to whether or not a connection exists between each Master and Slave. If all the Slaves in a SCR can receive requests from all the Masters, the SCR is said to be 'Fully Connected'. In this analysis, in addition to the variation of parameters seen in the previous chapter, we vary the Connectivity of the SCR.

For this main SCR, 'Full Connectivity' implies all the Slaves can receive requests from all the Masters. That implies each of the fifteen Slaves is connected to Sixteen Masters. 'Reduced Connectivity' implies, that a Slave can receive requests from a certain set of Masters. In this main SCR, the Reduced Connectivity configuration is such that:

- Four slaves are connected to 11 masters each,
- Four other slaves are connected to four other masters,
- Five slaves are connected to 16 masters, and
- Two slaves are connected to 15 masters each

In the below analysis, the default arbitration scheme for the read requests is Dynamic Round Robin Fixed Priority (DRRP). In cases where rr-read is specified, it implies that the arbitration scheme for the read requests is Round Robin.

A SUMMARY OF THE ANALYSIS:

In this part we will discuss on the different experiments, which are conducted to determine a sweet spot of the SCR architecture. This analysis helps us to understand on how best to judge and to determine the best architecture configuration for a given set of specifications/parameters. In this analysis, the priority to determine the Sweet spots for individual cases is in the following order Performance, Power, and Area. The priorities vary across the industry based on the needs.

Specification/Parameters

The parameters used for this analysis are as follows:

Protocol = VBUSM	Temperature = -40ºC
Number of Masters = 16	Number of pipeline stages = variable
Number of Slaves = 15	Data path width = 128 bits
Frequency of Synthesis = 500MHz	Arbitration = Variable
Voltage Domain = Variable	Connectivity: Full and Reduced

Connectivity Analysis:

An analysis on the Full and Reduced Connectivity seen on Area, Power and Path Slack with other parameters kept constant.

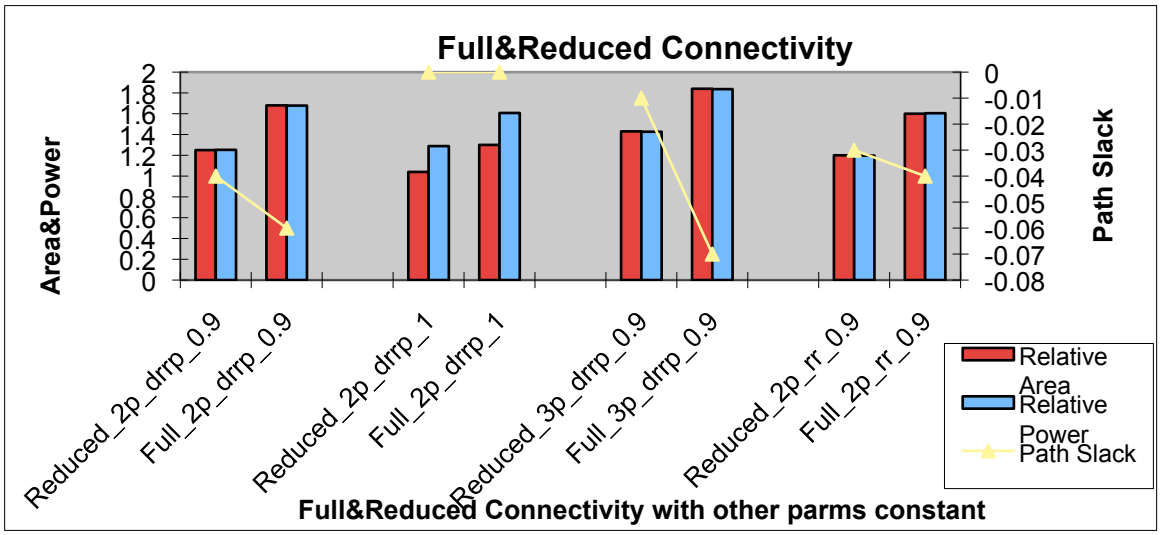

Figure 36: Connectivity analysis of main SCR

Sweet Spot: In reference to Figure 36, Reduced_2p_drrp_1V is the sweet spot. Reduced_2p_drrp_1V is the configuration with reduced connectivity, 2 pipeline stages, Dynamic Round Robin Fixed Priority arbitration technique and at a voltage supply of 1V.

Pipeline Depth Analysis with Full Connectivity:

An analysis on the full connectivity seen on Area, Power and Path Slack with other parameters kept constant.

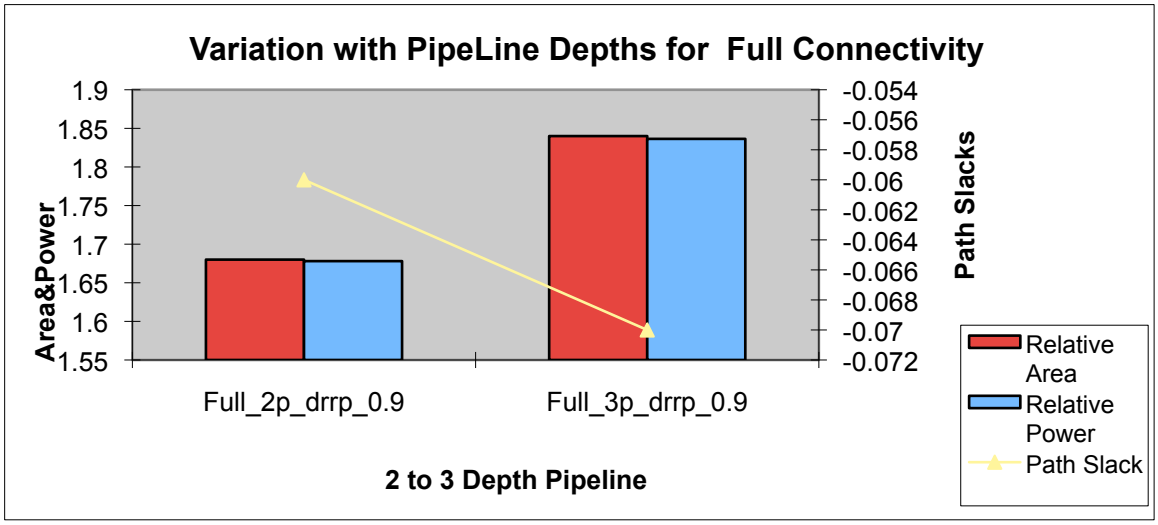

Figure 37: Pipeline variation depth analysis of main SCR

Sweet Spot: In reference to Figure 37, Full_2p_drrp_0.9 is the sweet spot. Full_2p_drrp_0.9 is the configuration with full connectivity, 2 pipeline stages, Dynamic Round Robin Fixed Priority arbitration technique and at a voltage supply of 0.9V.

Exception: The path slack did not improve significantly with increase in the depth of pipelines, instead degraded the area and power by around 10%. Here we would have expected the pipeline stages to achieve the improved performance.

Pipeline Depth Analysis with Reduced Connectivity:

An analysis on the reduced connectivity seen on Area, Power and Path Slack with other parameters kept constant.

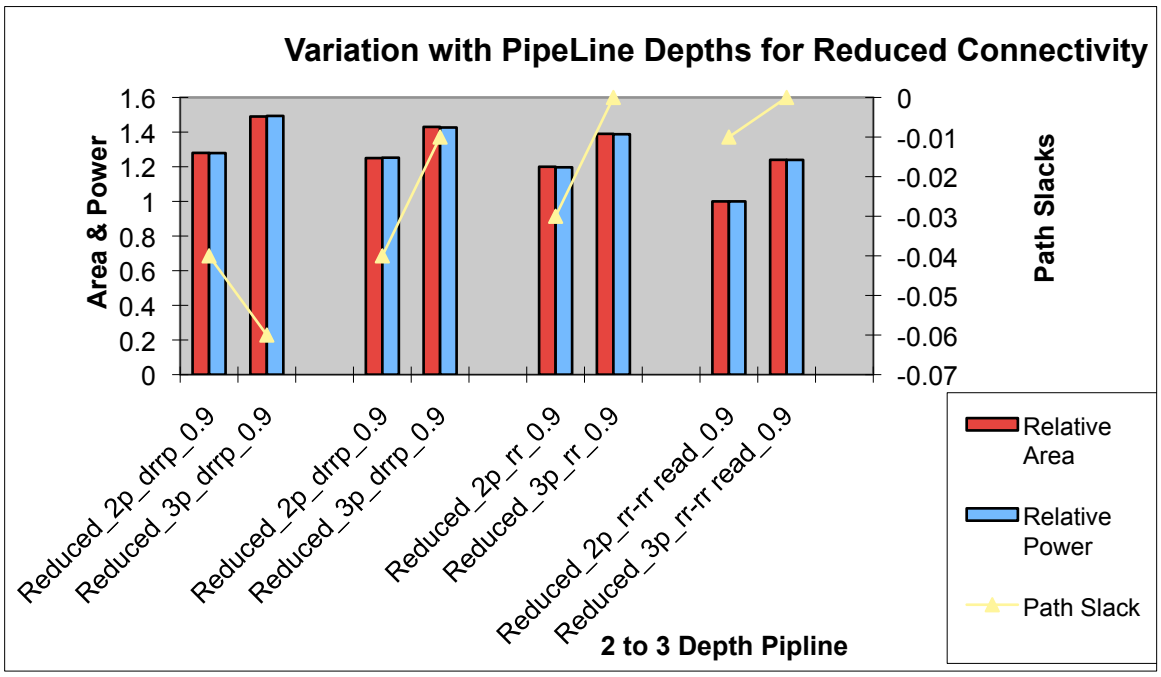

Figure 38: Pipeline depth analysis, with reduced connectivity of main SCR

Sweet Spot: In reference to Figure 38, Reduced_2p_rr_rr-read_0.9 is the sweet spot. Reduced_2p_rr_rr-read_0.9 is the configuration with reduced connectivity, 2 pipeline stages, Round Robin arbitration technique for read and write requests and at a voltage supply of 0.9V.

Arbitration Analysis:

An analysis on the full and reduced connectivity seen on Area, Power and Path Slack with other parameters kept constant.

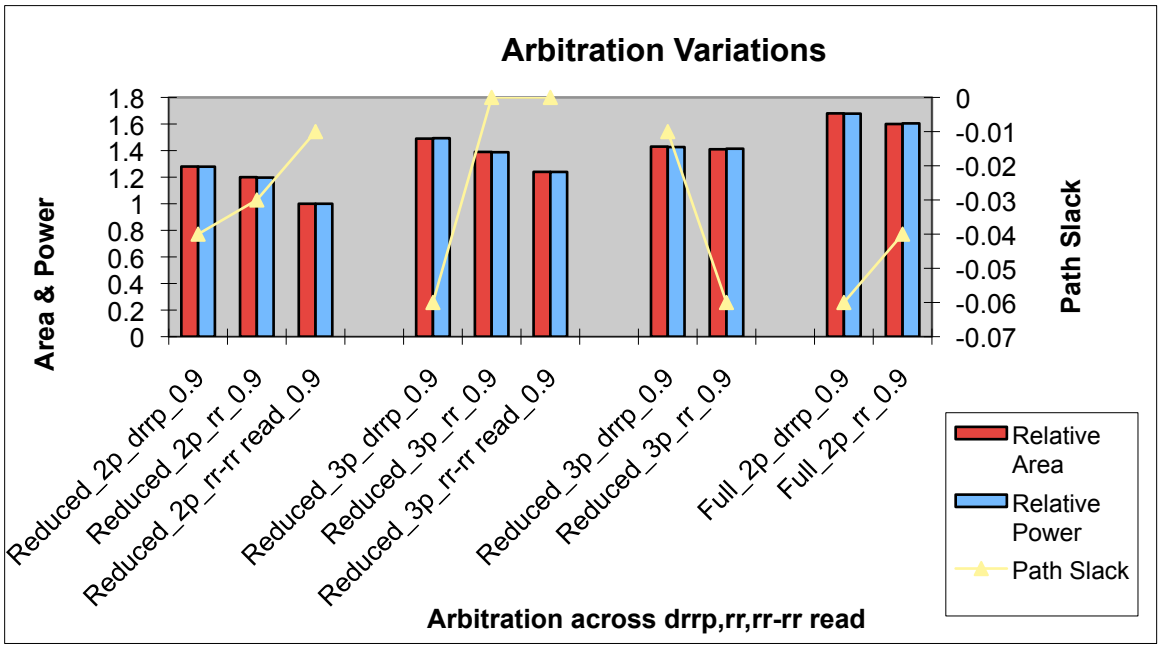

Figure 39: Arbitration variation analysis of main SCR

Sweet Spot: In reference to Figure 39, Reduced_2p_rr_rr-read_0.9 is the sweet spot. Reduced_2p_rr_rr-read_0.9 is the configuration with reduced connectivity, 2 pipeline stages, Round Robin arbitration technique for read and write requests and at a voltage supply of 0.9V.

Voltage Analysis:

An analysis on the full and reduced connectivity seen on Area, Power and Path Slack with other parameters kept constant, with varying voltage supply.

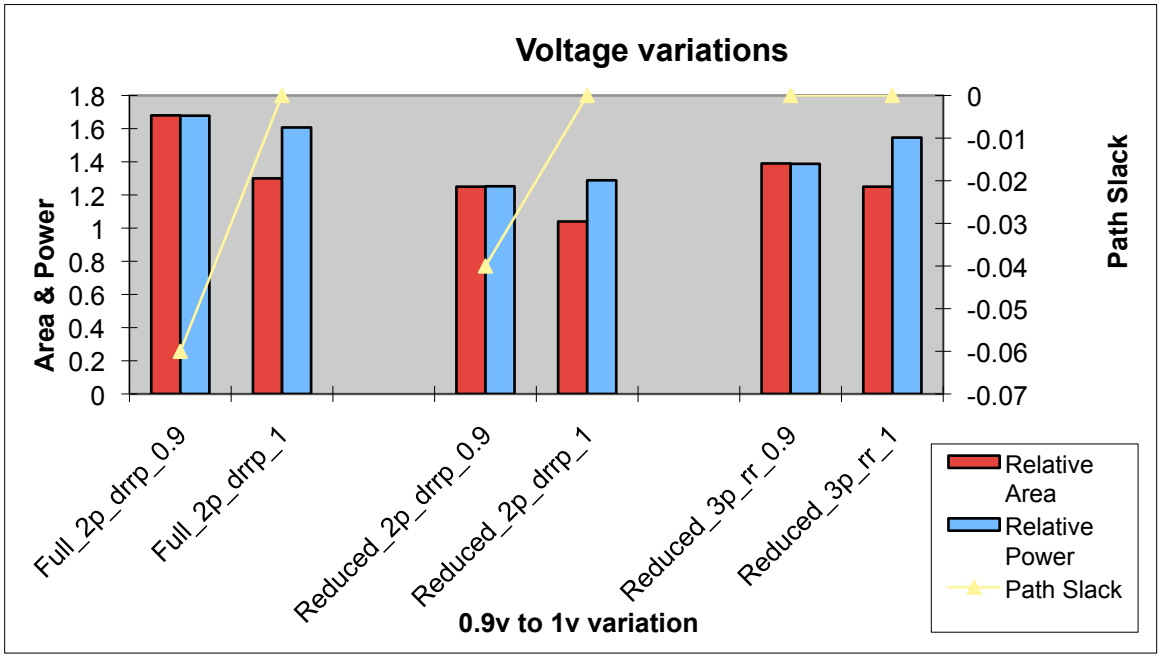

Figure 40: Voltage variation analysis of main SCR

Sweet Spot: In reference to Figure 40, Reduced_2p_drrfp_1V is the sweet spot. Reduced_2p_drrfp_1V is the configuration with reduced connectivity, 2 pipeline stages, Dynamic Round Robin Fixed Priority arbitration technique and at a voltage supply of 1V.

Exception: The Power and Area increased with decrease in voltage. This effect is caused due to the additional logic, required for timing closure. Here the increase in area dominated the decrease in voltage.

Overall Analysis to obtain a close to ideal Sweet spot architecture to meet the specification:

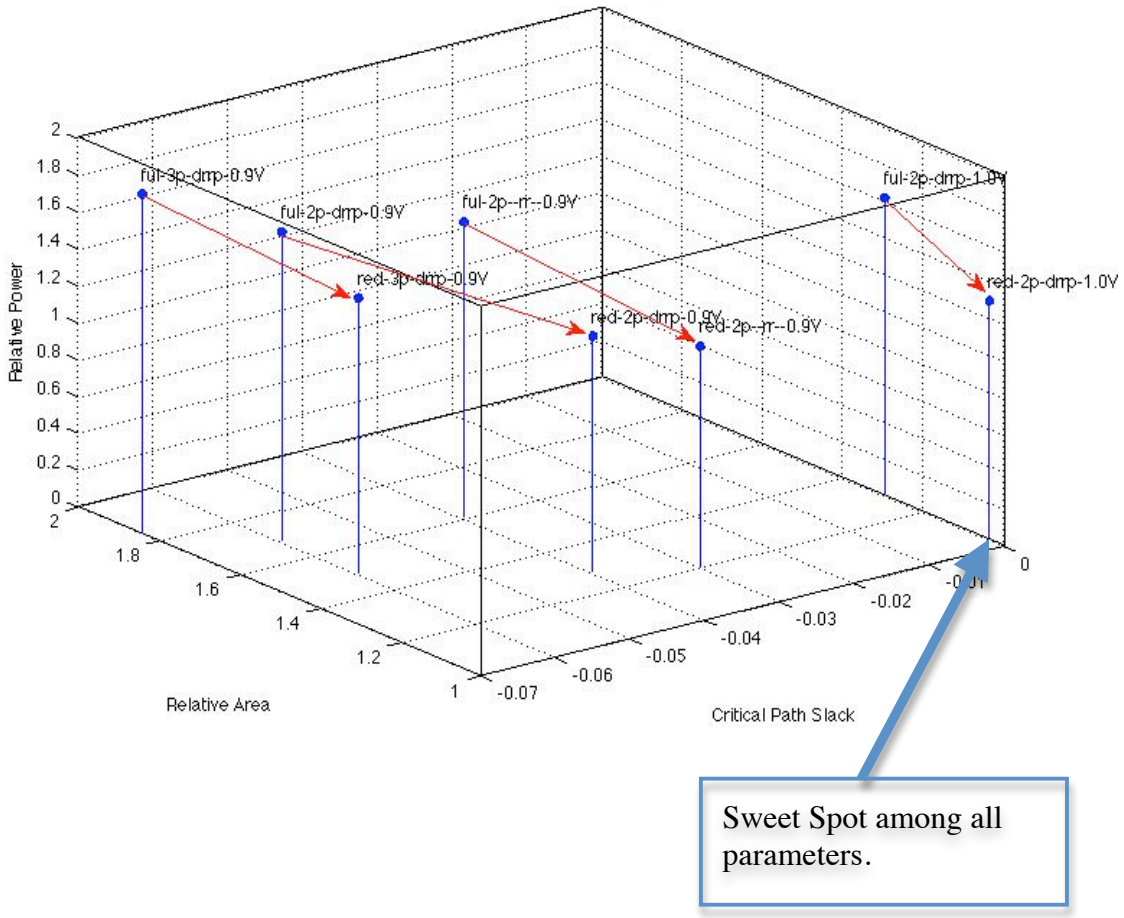

Figure 41: Sweet spot detection main SCR

This chapter provided an overview to decide an architectural sweet spot for the main SCR configuration used in the Texas Instruments subsystem. Figure 41 depicts the final sweet spot for this SCR with the aid of individual sweet spots used from independent analyses (Voltage/Connectivity/Pipelines variation etc.) explained earlier. The sweet spot is the configuration with minimum power consumption, minimum area and minimum critical path slack among the various architectural configurations. The think arrow in Figure 41 indicates the sweet spot.

Chapter 6: Classification and Analysis of Deadlocks in a Distributed SCR Network

Deadlock is a condition in which the throughput of a network or part of a network goes to zero due to conflicts in resource acquisition. In other words, a network is in deadlock if the network resources are suspended waiting for each other. There does exist work regarding the analysis and solutions to avoid the various kinds of deadlocks in networks that consist of a single SCR [5, 6].

In the current complex SoC Bus Architectures, multiple SCRs are interconnected with or without bridges to obtain the required performance and cost metrics. This is a Distributed SCR network [8, 12]. The part of the SCR network which interfaces high performance units, and locations where high performance is required; multiple issue bus protocols are used. In such protocols (such as the VBUSM protocol or AXI protocol), which allow multiple transactions to be outstanding implemented by transaction pipelining, it drastically increases the system performance. In the VBUSM protocol, for write requests, the master sends out multiple requests (address requests), but sends the write data in the same order of the requests, after the slave has acknowledged the previous request. The read requests do not need an acknowledgement.

In such a distributed network of SCRs, which use multiple issue protocols (such as VBUSM or AXI), new kinds of deadlocks occur. One such deadlock class in if there are parallel paths to a single slave. In this chapter we will classify and analyze all such deadlocks. Mitigations to these deadlocks are provided with respect to the Texas Instruments SCRs, but the same principles can be applied in other such networks.

ADDITIONAL TERMINOLOGY

Top-Level Slave

A Slave, which emerges out from the entire Distributed SCR network being treated as one component, is a Top-Level Slave. It does not further connect to other SCRs.

Top-Level Master

A Master, which acts as a Master to the entire Distributed SCR network being treated as one component, is a Top-Level Master. It does not originate from another SCR in this network.

Local Slave

A Slave on a SCR is said to be local to a Master on the same SCR, if it terminates as a top-level slave and does not further connect to another SCR.

External Slave

A Slave on a SCR is said to be external to a Master on the same SCR, if it does not terminates as a top-level slave and but is further connect to another SCR in the network.

CLASS 1: DEADLOCKS WITH PARALLEL PATHS TO A SINGLE SLAVE

Class 1 deadlocks are mechanisms where it is possible for two write commands from a master to take different paths to the same slave, and with this architecture its possible for the second command to surpass the first.

Case 1: Single master to single slave

Case 1 deadlock is defined situations where the deadlock arises due to parallel paths originating from a single master to a single slave.

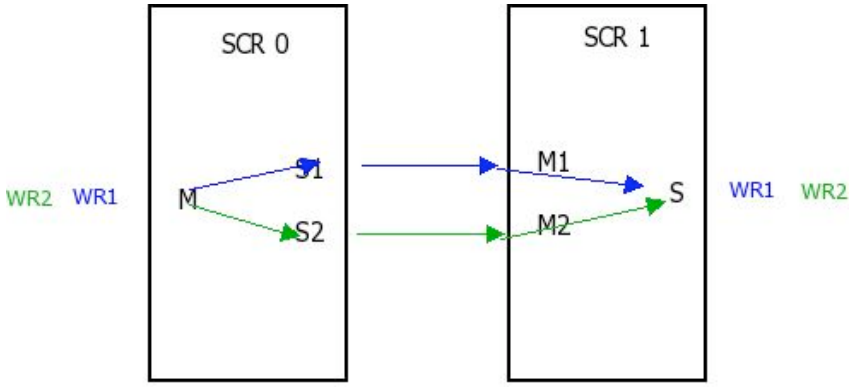

Figure 42: Case 1 Deadlock – same master through two paths to a slave.

Analysis: Possible if paths exist. It's possible that the master has write data for the first command available when the slave is expecting the write data for the second command. This first write data is not acknowledged. In Figure 42, the master sends WR1 earlier than WR2. At S1 and S2 ports in SCR0, WR1 arrives earlier than WR2. However, if Slave in SCR1 is busy when WR1 arrives, WR1 will have to wait. When WR2 arrives to the slave port and if M2 has a higher priority than M1, slave receives WR2 earlier than WR1.

Mitigation: The Subsystem includes mechanisms to prevent this deadlock at the source. The system has multiple paths from masters to some slaves. However the design is prevented by hardware to have at any time only a single path available to a given master to a given slave.

This mechanism is provided by a combination of select bits in a control register, which in turn adjusts the address for masters to force them to select a single path to a slave, regardless of the original address.

Case 2: 2 masters and 1 slave

Case 2 deadlocks are defined as a race condition between 2 masters and 1 slave, with a common infrastructure point among the paths.

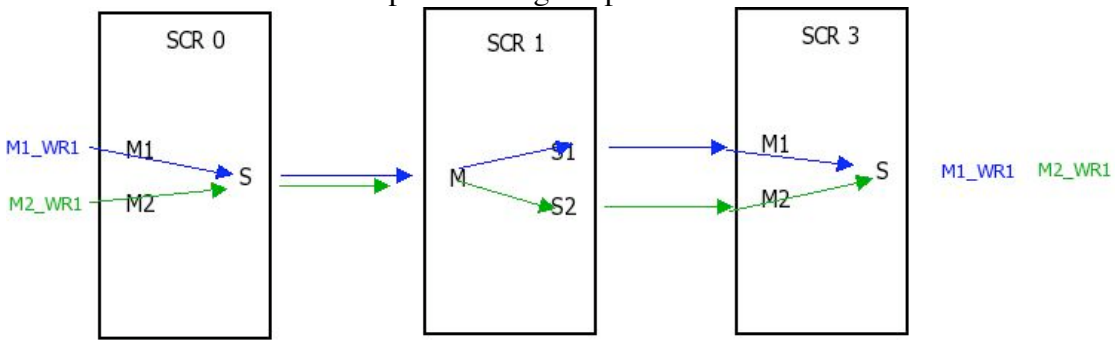

Figure 43: Case 2 Deadlock – two different masters, combined to a single segment, through two paths to a slave.

Analysis: it is possible for two write commands to take the same path to the same slave, and with this architecture its possible for the second command to surpass the first. In these situations it's possible that the common infrastructure point has write data for the first command available when the slave is expecting the write data for the second command. This first write data is not acknowledged. In figure 43, M1_WR1 was send and received first by SCR0, slave S, followed by M2_WR1. SCR2 has both the requests on its masters and services M2_WR1 first.

Mitigation: The Subsystem includes mechanisms to prevent this deadlock at the source. The system has multiple paths from masters to some slaves. However the design is prevented by hardware to have at any time only a single path available to a given master to a given slave.

This mechanism is provided by a combination of select bits in a control register, which in turn adjusts the address for masters to force them to select a single path to a slave, regardless of the original address.

CLASS 2: DEADLOCKS WITH PARALLEL PATHS TO TWO OR MORE SLAVES

Class 2 deadlocks are mechanisms where there is a race condition between 2 or more masters and 2 or more slaves.

Case 3: 2 Masters and 2 slaves on different bus segments

Case 3 deadlock is defined as a race condition between 2 masters and 2 slaves on two separate bus segments.

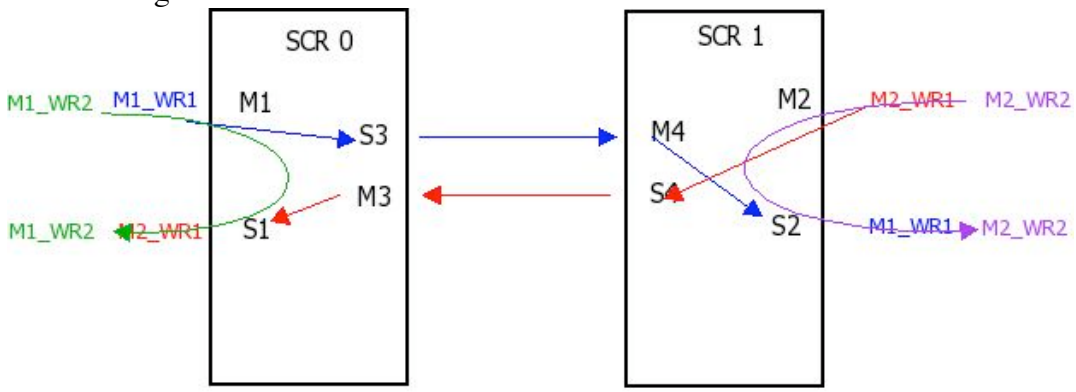

Figure 44: Case 3 Deadlock – between 2 Masters and 2 Slaves.

***Analysis*:** This deadlock is possible.

In Figure 44, It is possible that both masters issue two writes "WR1" which are accepted by the infrastructure, but not the final destination slaves. Subsequently both issue "WR2" transactions to a local slave on the segment where the masters reside. It is possible that both of these "WR2" transactions are accepted by the slaves before the "WR1" transactions are. The deadlock occurs then when the slaves are expecting the

WR2 data from the masters, but the masters are waiting for the WR1 data to be acknowledged.

Mitigation: The TI in-house SCR has built in mechanisms to protect against this kind of deadlock situation. The SCRs classify slaves as 'external' if they are connected through subsequent infrastructure and 'local' if the slave is directly connected to the SCR. Writes to 'local' slaves are prevented from surpassing the writes to 'external' slaves (stopped before the arbiter) until the master has provided the write-data for the external request. At this point the local transaction can proceed. External transactions are not held up for local transactions.

Case 4: Parallel paths from 2 or more masters to 2 or more slaves

Case 4 deadlock is an extension of Case 3, but where the slaves are both external (or local depending on the SCR segment), so the ext/local ordering delays are not applicable.

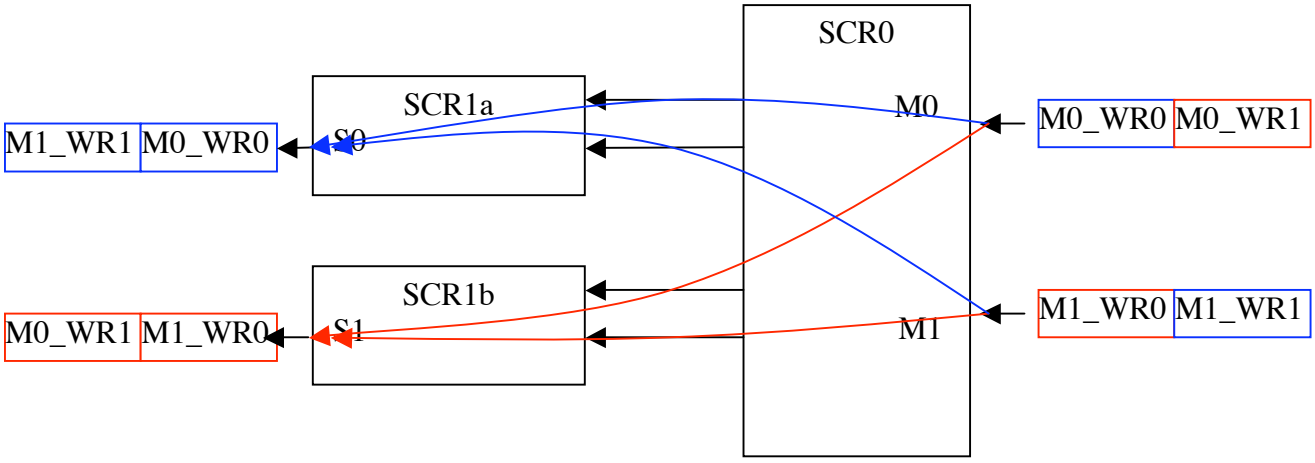

Figure 45: Case 4 Deadlock – parallel paths from 2 or more Masters to 2 or more Slaves

Analysis: The analysis for this specific case is that this *condition is possible*. This issue is related to the pair of masters from SCR0 to the SCR1 clusters. The address translate logic corrects the multi-command path (deadlock race condition) from masters to slaves taken 1 at a time. The issue identified is the issue of deadlock race conditions when masters and slaves are taken 2 at a time. It requires that there be a pair of parallel paths, where the pairs of commands can pass each other.

Please refer to the Figure 46 for analysis details. The circles represent arbitration points. The small rectangles represent pipeline stages. Lines in black are points where commands execute serially. Lines in blue are where commands can operate in parallel. In essence if there is a parallel command path, it is assumed that previous bus activity could cause the masters CMD0's to be delayed to arrive after the CMD1's, causing the deadlock where no slaves are requesting the master WD0's.

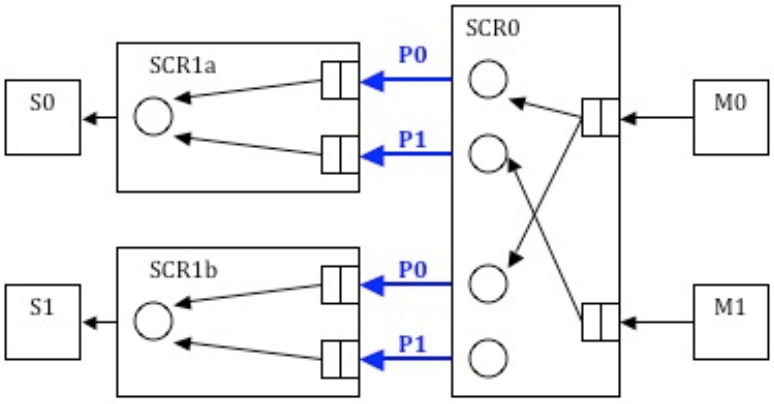

Figure 46: Case 4 Deadlock Analysis– two different masters, combined to a single segment, through two paths to a slave.

Mitigation: Mitigation involves either removing the hazard in the topology (items 1 and 3 below) or adding a mechanism to re-align to eliminate the hazard (items 2 and 4 below).

Some ways to mitigate this race condition are listed below. (Others may exist.)

1. Select the P0/P1 from the SCR0 based on slave decoding, not for each master.

   Specifically, each slave will determine which port the transaction will use. For example, the M0 slave could be addressed on P0, while M1 on P1. In this manner, the SCR0 address translation logic will not be required, see Figure 47. This is the lowest risk solution. In this manner, from any master to any slave will have 1 and only 1 path. This option has the benefits of solving the deadlock condition as well as removing the other address translation complexities.

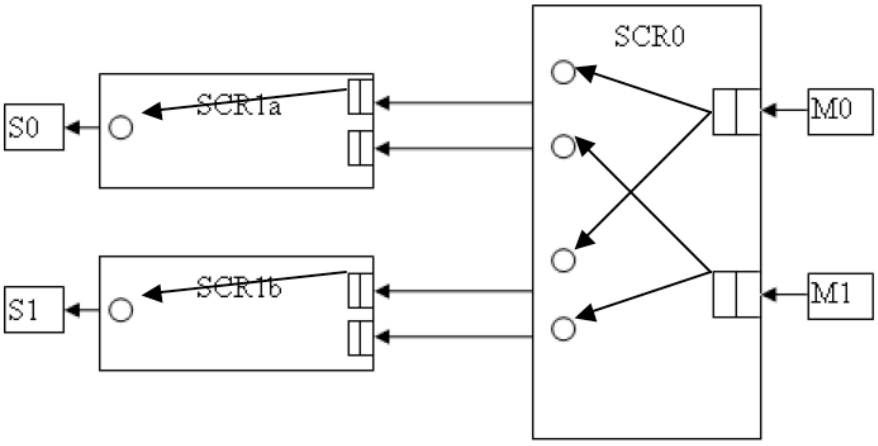

Figure 47: Case 4 Deadlock Mitigation – two different masters, combined to a single segment, through two paths to a slave.

2. Introduce a store-forward bridge between SCR0 and SCR1

   Introducing an asynchronous bridge on one of the paths will break up the deadlock condition. In this manner one of the parallel paths will be broken, as the data will be required to 'catch up with' the command. This method has the impact to add latency on both writes and reads across the path where the bridge is added.

3. Change the hierarchy so only a single path from each SCR exists.

   Another option is to introduce a hierarchy change to remove the hazard.

4. Update to SCR to delay between any external transactions.

   Modify the SCR internals to stall between external transactions to allow for commands and write data to catch up, similar to condition between external to local transactions. Doing so, will reduce the number of parallel transactions that can be issued.

Case 5: Parallel paths from 2 or more masters to 2 or more slaves

Case 5 deadlocks appears to be an extension of Case 3, but is not really a deadlock condition.

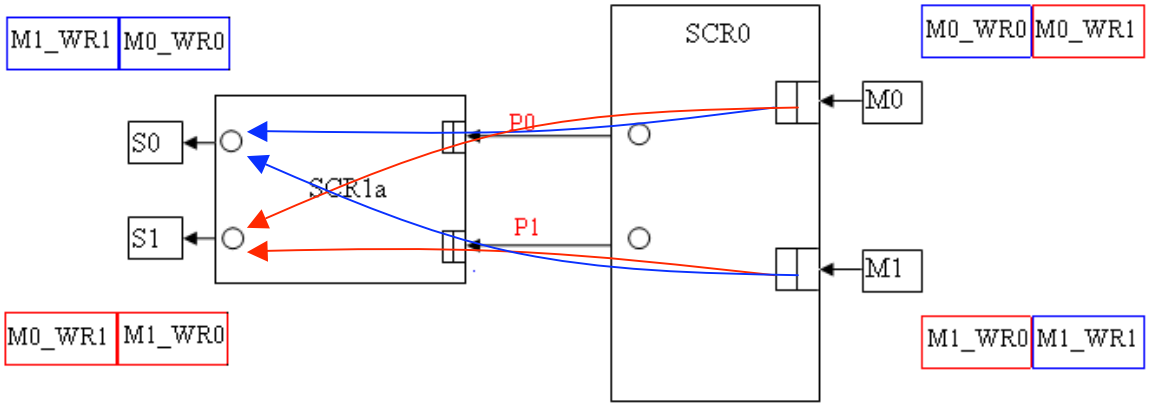

Figure 48: Case 5 Deadlock – parallel paths between 2 or more Masters to 2 or more Slaves.

Analysis: The analysis for this specific case is that this condition *is not possible*. There is not a parallel path in this situation; rather all the paths are sequential.

Please refer to the Figure 49 for analysis details. The circles represent arbitration points. The small rectangles represent pipeline stages. Lines in black are points where commands execute serially. Lines in blue are where commands can operate in parallel.

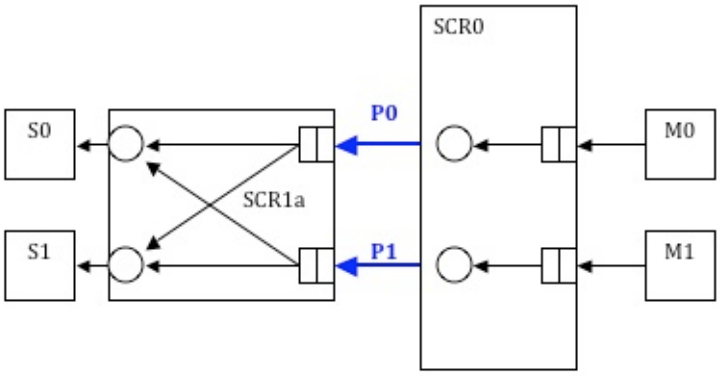

Figure 49: Case 5 Deadlock Analysis – parallel paths between 2 or more Masters to 2 or more Slaves.

Case 6: Parallel paths from 2 masters to 2 slaves

Case 6 deadlocks are similar to Case 4, parallel paths from 2 masters to 2 slaves through 3 segments

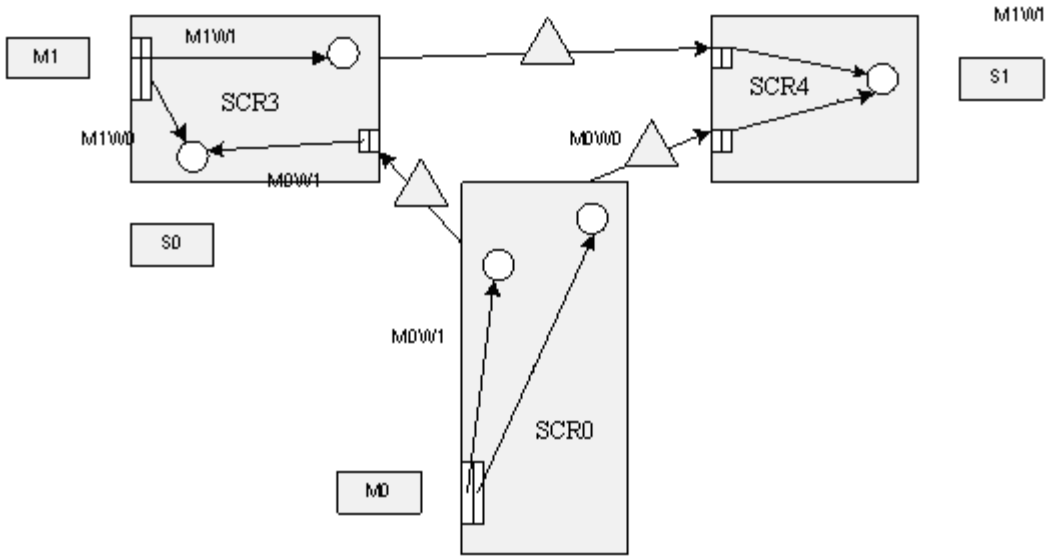

Figure 50: Case 6 Deadlock – parallel paths between 2 Masters to 2 Slaves.

Analysis: The analysis for this specific case is that this condition *is possible*. Matching pair of parallel paths exists to allow the commands to pass each other. This is very unlikely, but cannot be proven impossible. It's unlikely that there would be such a slow bridge that would allow the M1W1 to pass as needed for the deadlock condition.

Mitigation: Mitigation involves adding a mechanism to re-align to eliminate the hazard (items 1 and 2 below).

1. Introduce a store-forward bridge between SCR0 and SCR3

   Introducing an asynchronous bridge on the paths between SCR0 and SCR3 will break up the deadlock condition. In this manner one of the parallel paths will be broken, as the data will be required to 'catch up with' the command. This method has the impact to add latency on both writes and reads across the path where the bridge is added.

2. Update to SCR to delay between any external transactions.

   Modify the SCR internals to stall between external transactions to allow for commands and write data to catch up, similar to condition between external to local transactions. This option is not recommended due to reduction in number of parallel paths thought which requests can be issued. This change also may include a high effort for verification.

CLASS 3: DEADLOCKS DUE TO OPERATION INTERRUPTION

Active Bus Exception

A further type of deadlock occurs when a transaction can be interrupted during the middle of processing a valid transaction. In the TI Subsystem design, there are two

types of cases where this exception is a potential issue – selective reset of IP and clock gating. The design by its construction prevents these types of deadlocks.

Clock Gating

The mechanism for preventing this deadlock is that a peripheral before getting clock gated acknowledges that all bus transactions it is processing are completed. This can be accomplished by a simple handshake.

When a peripheral is to have its clock stopped, a clock stop request is asserted. At this time, the bus infrastructure is informed to direct all subsequent commands to the null slave (default slave) for processing. When the peripheral has no further bus activity (when it is safe to clock gate) it responds with the clock stop acknowledge signal.

In this manner no transactions are interrupted, and deadlock is avoided.

Individual Peripheral Reset

The mechanism for preventing this deadlock is that a peripheral before getting reset follows the above outlined handshake process. A control module, when resetting a peripheral first uses the handshake process to complete any pending transactions. Once the handshake completes, the clock will be gated for the IP, then send out the reset signal. Clocks will resume while the reset is asserted and then the reset will be removed. Once the reset is removed the bus will be signaled to allow transactions to be processed by the peripheral.

CLASS 4: DEADLOCKS DUE TO PROTOCOL VIOLATION

Peripherals should not be permitted to behave in a manner, which violates the protocol.

Chapter 7: Algorithm to detect deadlocks in Distributed SCR Network

This chapter provides a sample-working algorithm to detect all possible deadlocks as mention in Chapter 6. For the case study, this algorithm is applied on the Texas Instruments wireless base station subsystem (NoC Interconnect). This algorithm works as an interface tool to help mitigate any architectural problems very early in the design cycle.

In the case study, this algorithm is used in developing the Interconnect for the TI Sub-system, from a given configuration - containing deadlock issues – to a refined deadlock free interconnect configuration.

OVERVIEW OF THE ALGORITHM

Nomenclature Rules

1. Top Level Masters, i.e. masters who are not linked to any other slaves from another SCR are prefixed with “SOURCE_”
2. Top Level Slaves, i.e. slaves who do not get linked to any other masters from other SCR are prefixed with “DEST_”

Phases of the Script

The first phase of the script is to generate connectivity files for each SCR. The algorithm reads a given configuration file of a single SCR, generates all the master and slave names used in the SCR into a file to be used for re-naming. The output file is modified to rename all the masters and slaves if needed according to the Nomenclature rules, and to obtain connectivity between different SCRs. For example if slave S0 of SCR0 needs to get connected to master M1 of SCR1, The connectivity statement “S0 S0_M1” needs to be specified in a connectivity file.

In the second phase, the algorithm takes in multiple connectivity files for each SCR, and combines them to form a top-level main connectivity file. Renaming is used in this level to connect slaves from one SCR to masters of another SCR. In this phase it generates the connectivity's at top level, i.e. eliminating the SCR boundaries, by replacing each master with the details of all other masters from other SCRs which get connected to it, till the top level masters are reached. The outputs of this phase include the connectivity file at top level, removing the hierarchy.

In the third phase, this algorithm reads in the main connectivity file, which is the output of the previous phase. The previous phases can also be skipped if the main connectivity file already exists. If the main connectivity list at top level has the same master connected to a particular slave more than once, it implies that there are more than one ways from the master to the slave, revealing the parallel paths from a single master to a single slave. This phase also generates a route file, which contains the path from the masters to the slaves. If the result of this phase, determines that there are parallel paths, the analysis should stop and a reconfiguration of the SCR network is required.

The final phase of this algorithm finds deadlock conditions between multiple masters and multiple slaves. This phase is to be executed only if parallel paths between a single master and single slave do not exist, if any are there, it would have been found in the pervious phase.

Inputs required for this phase are:

- Initial connectivity file: output of phase 2
- Top level connectivity file: output of phase 3

- Masters list: contains a list of all the masters you want to send instructions through to test for deadlocks, should contain all the masters for exhaustive testing.
- Number of masters at a time: The number of masters to be used at a time from the masters list to send out instructions, should be equal to the total number of masters for exhaustive testing.
- Number of instructions to be sent from each master
- Number of slaves: Maximum number of slaves all the instructions can go to, for exhaustive this number should be less than or equal to (Number of masters at a time * Number of instructions per master)- 2. There would be redundancy otherwise, as is all other cases, there is no condition for dead lock to be possible
- Route file: which is the output of phase 3
- External slave file: which contains a list slaves configured as external slaves

The output produced from this phase: Log file, Sample below

```
########################################################################
 ANALYSING LOCK WITH NO PARALLEL PATHS FROM SAME MASTER TO SAME SLAVE
Content of BIN 0 is SOURCE_SCR3_M0:1 SOURCE_SCR3_M1:0
Content of BIN 1 is SOURCE_SCR3_M1:1 SOURCE_SCR3_M0:0
DEST_SCR4_S0,DEST_SCR10_S0 -> DEAD LOCK POSSIBLE, EXT-EXT
DEST_SCR4_S0,DEST_SCR10_S0 -> DEAD LOCK POSSIBLE, EXT-EXT
```

- In the log: BIN == SLAVE 0 hence Content of BIN 0 implies the instructions which would have been received by SLAVE 0, Left side received earlier than right.
- SOURCE_SCR3_M1:0 => Master M1 of the SCR3 SCR 's 0^{th} Instruction
- DEST_SCR4_S0,DEST_SCR10_S0 -> DEAD LOCK POSSIBLE, EXT-EXT => Dead lock is possible if S0 slave of SCR4 scr is SLAVE 0 and S0 slave of SCR10 SCR is slave 1, and this is a EXT EXT slave pair which is causing the deadlock.

ALGORITHM

The algorithm, for the main phase i.e. phase 4 of the script is as follows:

1. Obtain combinations of all possible masters of required number from the masters' list.
2. For each of the above, obtain combinations of all possible ways of instruction order for all the instructions (Number of instructions = Number of masters * Number of instructions per master).
3. For each of the above, obtain all possible ways they can be split among the number of slaves that is all different ways the instructions list can be split among the total number of ways.
4. For each combination, find out if that combination can be accepted completely by the slave or is a dead lock pattern.
5. If the above pattern is a dead lock pattern, then find out all the combinations of slaves for each bin.
6. For each combination, see if the instruction causing the deadlock is possible to be issued in the order.

a. Get the path information of the instruction, which causes the deadlock, and also the instruction before this from the same master. Calculate the point from which these paths are different.
b. See if the instructions can actually propagate by looking at the property of local slave and external slave of these points of divergence.
c. If both the slaves are local, the instructions can be issued in that order.
d. If both are external, the instructions can be issued in that order.
e. If the first instruction were external followed by a local, the instructions would not be issued.
f. If the first is local followed by external, the instructions can be issued in this order
g. If they can be issued, check all other deadlocking instructions in this pattern, to make sure the complete pattern can occur
h. If they cannot be issued continue to get the next pattern.

7. If an instruction pattern is possible to be issued, check the paths of the deadlocking instruction – instruction to be received after the deadlocking instruction to the same slave, instruction before deadlock from the master of the deadlocking instruction - the instruction before this is the receiving slave. See if the instructions can actually overtake the one behind them by analyzing the path.
8. If the instructions can be received after the path analysis, then deadlock exists.

Configuration 1

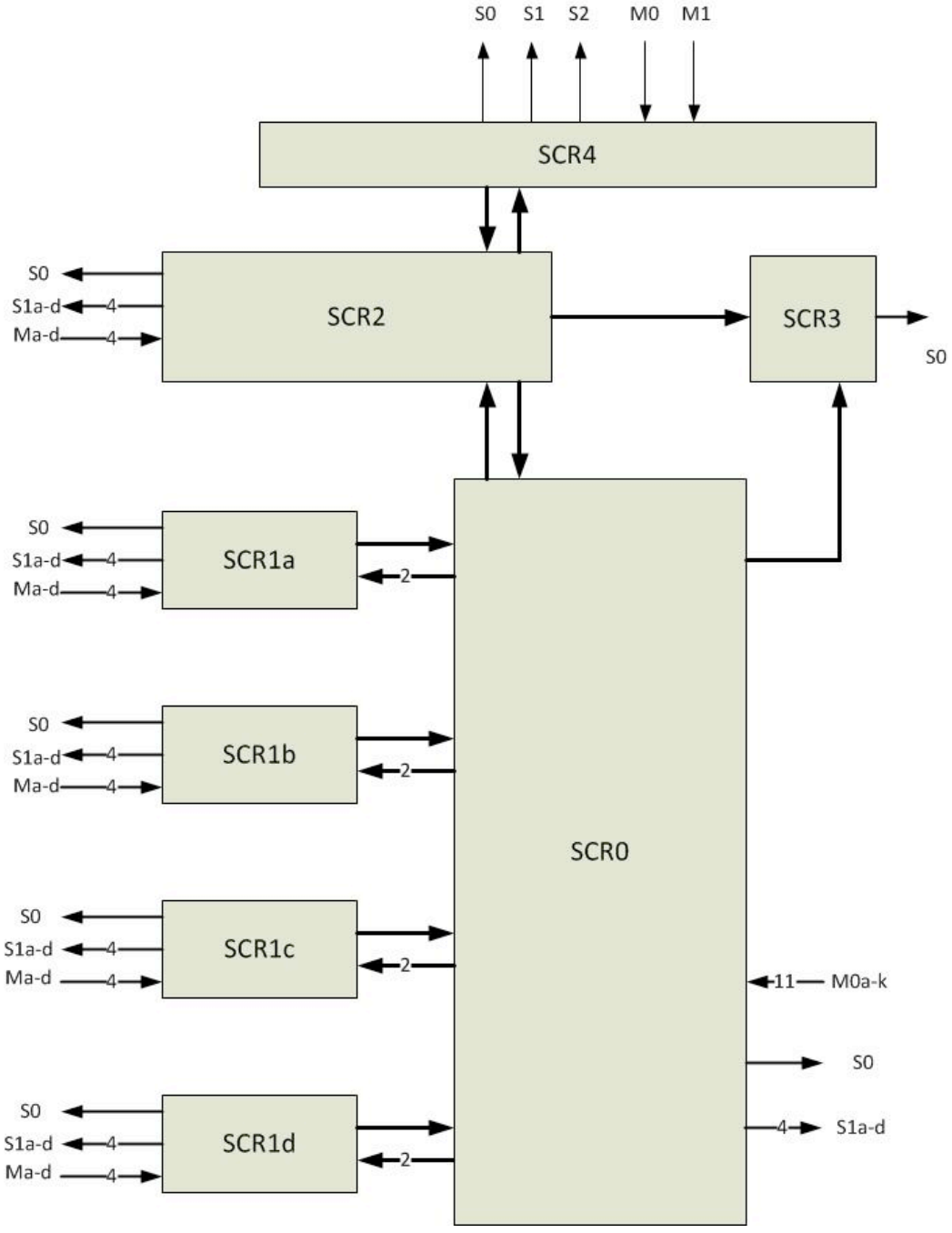

Figure 51: Configuration 1- Interconnect for TI Based Subsystem

Test Case used:

- All the deadlocks can be identified if there are at-least 2 top-level masters for each SCR, 2 local slaves and 2 external slaves.
- For the test to get a clear understanding, the numbers of local slaves in each SCR are minimized to 2 if there are more than 2 in the original spec. The number of top-level masters is reduced to 2 in each SCR if there are more than 2. And the SCR1 clusters were limited to 2. These limitations greatly reduce the redundant entries in the log files.

The connectivity's of the subsystem is as follows:

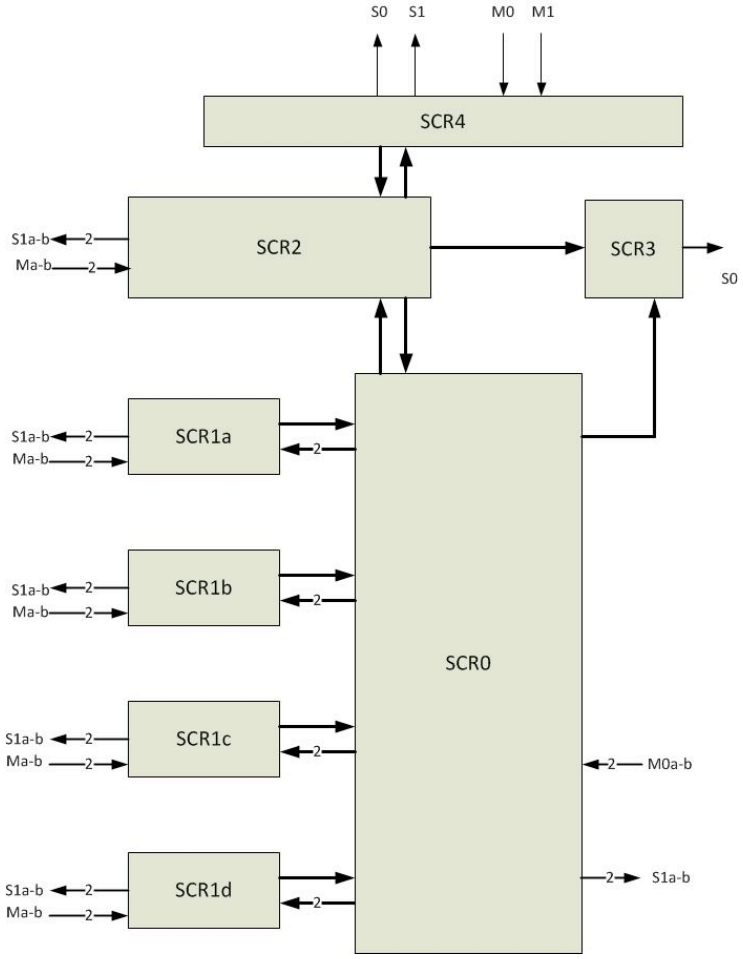

Figure 52: Configuration 1- Test case Interconnect for TI Based Subsystem

Analysis:

1. A SCR is fully connected if every slave in the SCR has a possible connection from every master in the SCR, with the exception of all masters, which are connected to the SCR the slave in consideration is connected to.
2. If SCR0 is fully connected
    a. Since there are 2 ports to each SCR1 (a-d) from SCR0, parallel paths would exist from any master in SCR0 to any slave in SCR1 (a-d)
    b. There would also be parallel paths to S0 slave of SCR3 from any path which passes through SCR0, (path 1: SCR0-> SCR3, path 2: SCR0-> SCR2-> SCR3)
3. Deadlock condition Case4 occurs.
4. To mitigate the deadlock:
    a. Case4-Mitigation1: Select between the 2 ports from the SCR0 based on slave decoding, not for each master.
    b. Using the following connectivity of SCR0

SLAVES	CONNECTED MASTERS					
SCR0_SCR1a_0	SCR1b_SCR0	SCR1c_SCR0	SCR1d_SCR0			
SCR0_SCR1a_1	SOURCE_SCR0_M0a	SOURCE_SCR0_M0b				
SCR0_SCR1b_0	SCR1a_SCR0	SCR1c_SCR0	SCR1d_SCR0			
SCR0_SCR1b_1	SOURCE_SCR0_M0a	SOURCE_SCR0_M0b				
SCR0_SCR1c_0	SCR1a_SCR0	SCR1b_SCR0	SCR1d_SCR0			
SCR0_SCR1c_1	SOURCE_SCR0_M0a	SOURCE_SCR0_M0b				
SCR0_SCR1d_0	SCR1a_SCR0	SCR1b_SCR0	SCR1c_SCR0			
SCR0_SCR1d_1	SOURCE_SCR0_M0a	SOURCE_SCR0_M0b				
DEST_SCR0_S1a	SCR1a_SCR0	SCR1b_SCR0	SCR1c_SCR0	SCR1d_SCR0	SOURCE_SCR0_M0a	SOURCE_SCR0_M0
DEST_SCR0_S1b	SCR1a_SCR0	SCR1b_SCR0	SCR1c_SCR0	SCR1d_SCR0	SOURCE_SCR0_M0a	SOURCE_SCR0_M0
SCR0_SCR2	SCR1a_SCR0	SCR1b_SCR0	SCR1c_SCR0	SCR1d_SCR0		
SCR0_SCR3	SOURCE_SCR0_M0a	SOURCE_SCR0_M0				

c. To eliminate deadlocks of other cases, we make the following connections as external

   i. SCR1a_SCR0
   ii. SCR1b_SCR0
   iii. SCR1c_SCR0
   iv. SCR1d_SCR0
   v. SCR0_SCR2
   vi. SCR2_SCR0
   vii. SCR4_SCR2

Resulting Configuration 2

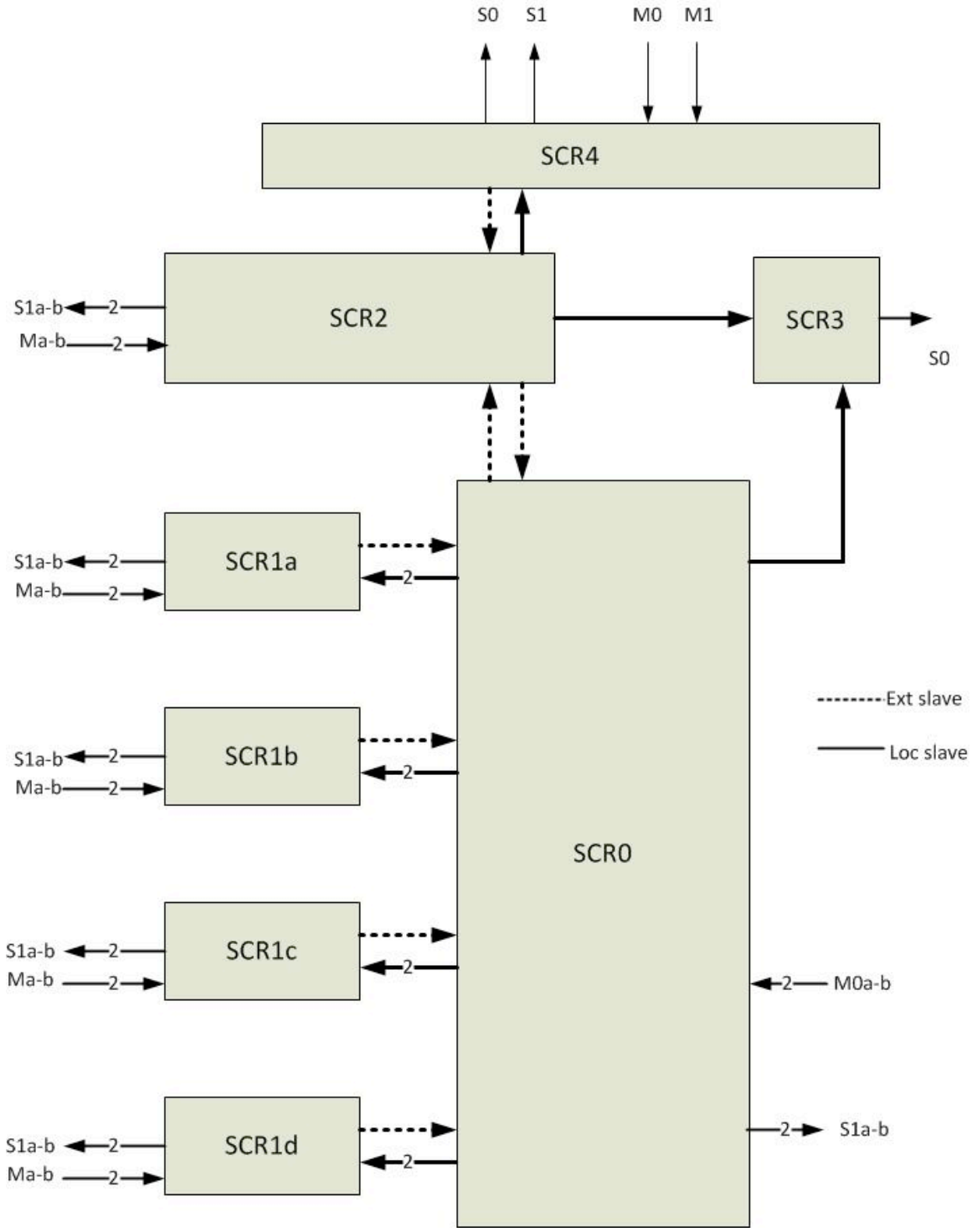

Figure 53: Configuration 2- Interconnect for TI Based Subsystem

a. REMAINING DEADLOCKS

Existing Deadlocks			
SCR which contains			
Master 0	Master 1	Slave 0	Slave 1
SCR0	SCR1a	SCR0	SCR1b
SCR0	SCR1a	SCR0	SCR1c
SCR0	SCR1a	SCR0	SCR1d
SCR0	SCR1a	SCR0	SCR3
SCR0	SCR1a	SCR1a	SCR1b
SCR0	SCR1a	SCR1a	SCR1c
SCR0	SCR1a	SCR1a	SCR1d
SCR0	SCR1a	SCR1a	SCR3
SCR0	SCR1a	SCR1b	SCR1c
SCR0	SCR1a	SCR1b	SCR1d
SCR0	SCR1a	SCR1b	SCR3
SCR0	SCR1a	SCR1c	SCR1d
SCR0	SCR1a	SCR1c	SCR3
SCR0	SCR1a	SCR1d	SCR3
SCR0	SCR1b	SCR0	SCR1a
SCR0	SCR1b	SCR0	SCR1c
SCR0	SCR1b	SCR0	SCR1d
SCR0	SCR1b	SCR0	SCR3
SCR0	SCR1b	SCR1a	SCR1b
SCR0	SCR1b	SCR1a	SCR1c
SCR0	SCR1b	SCR1a	SCR1d
SCR0	SCR1b	SCR1a	SCR3
SCR0	SCR1b	SCR1b	SCR1c
SCR0	SCR1b	SCR1b	SCR1d
SCR0	SCR1b	SCR1b	SCR3
SCR0	SCR1b	SCR1c	SCR1d
SCR0	SCR1b	SCR1c	SCR3
SCR0	SCR1b	SCR1d	SCR3
SCR0	SCR1c	SCR0	SCR1a
SCR0	SCR1c	SCR0	SCR1b
SCR0	SCR1c	SCR0	SCR1d
SCR0	SCR1c	SCR0	SCR3
SCR0	SCR1c	SCR1a	SCR1b

Existing Deadlocks			
SCR which contains			
Master 0	**Master 1**	**Slave 0**	**Slave 1**
SCR0	SCR1c	SCR1a	SCR1c
SCR0	SCR1c	SCR1a	SCR1d
SCR0	SCR1c	SCR1a	SCR3
SCR0	SCR1c	SCR1b	SCR1c
SCR0	SCR1c	SCR1b	SCR1d
SCR0	SCR1c	SCR1b	SCR3
SCR0	SCR1c	SCR1c	SCR1d
SCR0	SCR1c	SCR1c	SCR3
SCR0	SCR1c	SCR1d	SCR3
SCR0	SCR1d	SCR0	SCR1a
SCR0	SCR1d	SCR0	SCR1b
SCR0	SCR1d	SCR0	SCR1c
SCR0	SCR1d	SCR0	SCR3
SCR0	SCR1d	SCR1a	SCR1b
SCR0	SCR1d	SCR1a	SCR1c
SCR0	SCR1d	SCR1a	SCR1d
SCR0	SCR1d	SCR1a	SCR3
SCR0	SCR1d	SCR1b	SCR1c
SCR0	SCR1d	SCR1b	SCR1d
SCR0	SCR1d	SCR1b	SCR3
SCR0	SCR1d	SCR1c	SCR1d
SCR0	SCR1d	SCR1c	SCR3
SCR0	SCR1d	SCR1d	SCR3

Table 19: Deadlocks Configuration 2- Interconnect for TI Based Subsystem

Most of the above deadlocks exist, as there are 2 ports from SCR0 to SCR1 (a-d), creating possibilities for instructions to overtake.

Resulting Configuration 3

In addition to all modifications for configuration 2, in SCR1 (a-d) making each slave get requests either one of the ports form SCR0. The deadlocks, which remain, are listed in the Table 20.

Existing Deadlocks			
SCR which contains			
Master 0	**Master 1**	**Slave 0**	**Slave 1**
SCR0	SCR1a	SCR0	SCR3
SCR0	SCR1a	SCR1a	SCR3
SCR0	SCR1b	SCR0	SCR3
SCR0	SCR1b	SCR1b	SCR3
SCR0	SCR1c	SCR0	SCR3
SCR0	SCR1c	SCR1c	SCR3
SCR0	SCR1d	SCR0	SCR3
SCR0	SCR1d	SCR1d	SCR3

Table 20: Deadlocks Configuration 3- Interconnect for TI Based Subsystem

The above dead locks exists are there are 2 paths from SCR0 to slave S0 of SCR3, and both the paths having the possibility of a variable amount of delay from SCR0.

Mitigation

To mitigate the problem

1. Solution1: we can remove the SCR3, and split up its S0 slave into 2 slaves namely SCR3_S0a and SCR3_S0b, and use a logical module to combine the requests
2. Solution 2: Delay between external and external slaves.

Resulting Configuration 4a

In addition to configuration 3, eliminating SCR 3

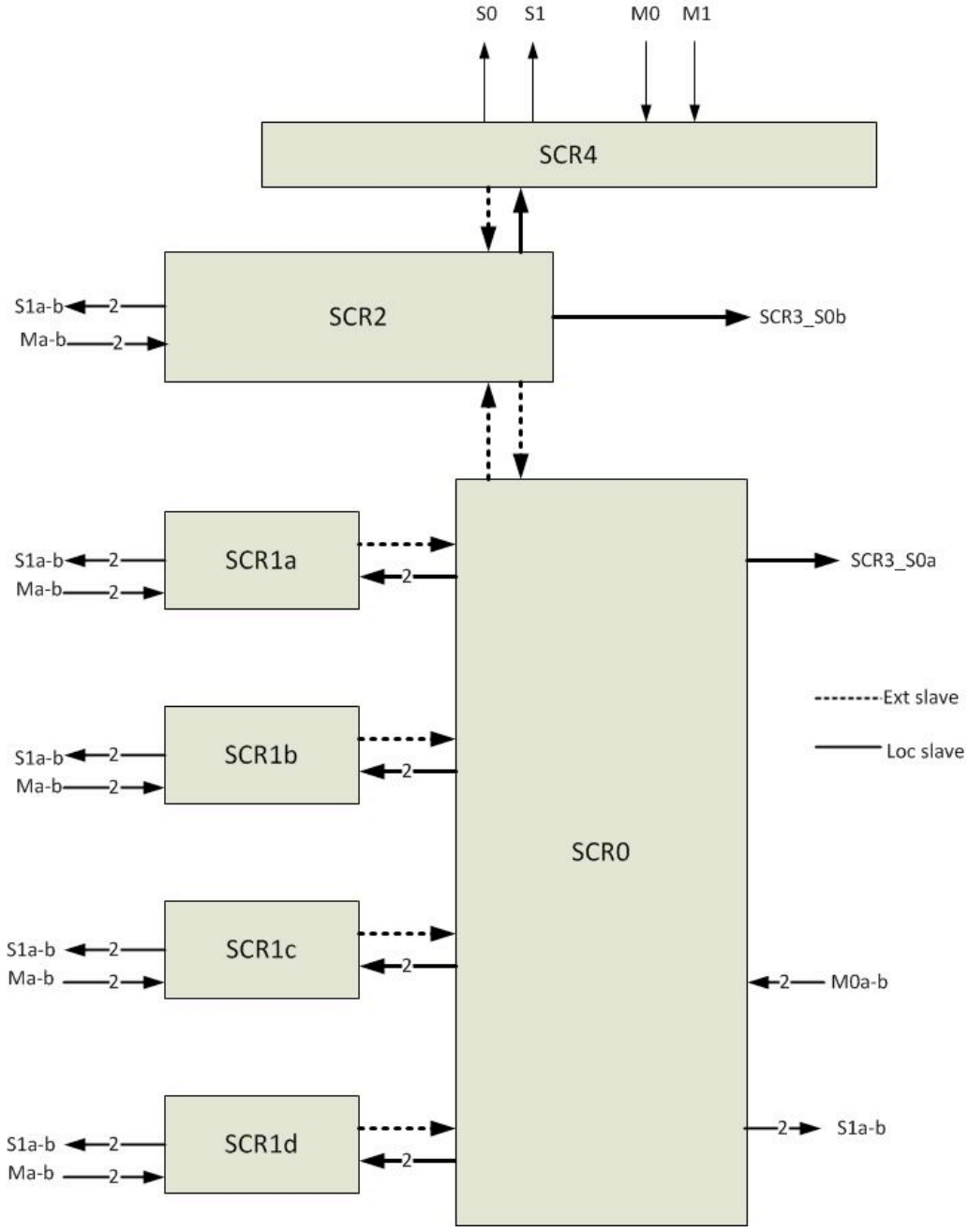

Figure 54: Configuration 4a- Interconnect for TI Based Subsystem

Analysis: No Deadlocks

Resulting Configuration 4b

In SCR1 (a-d) since each slave can get requests only from one port among the two from SCR0, eliminating the 2 ports and making it a single port from SCR0 to SCR1

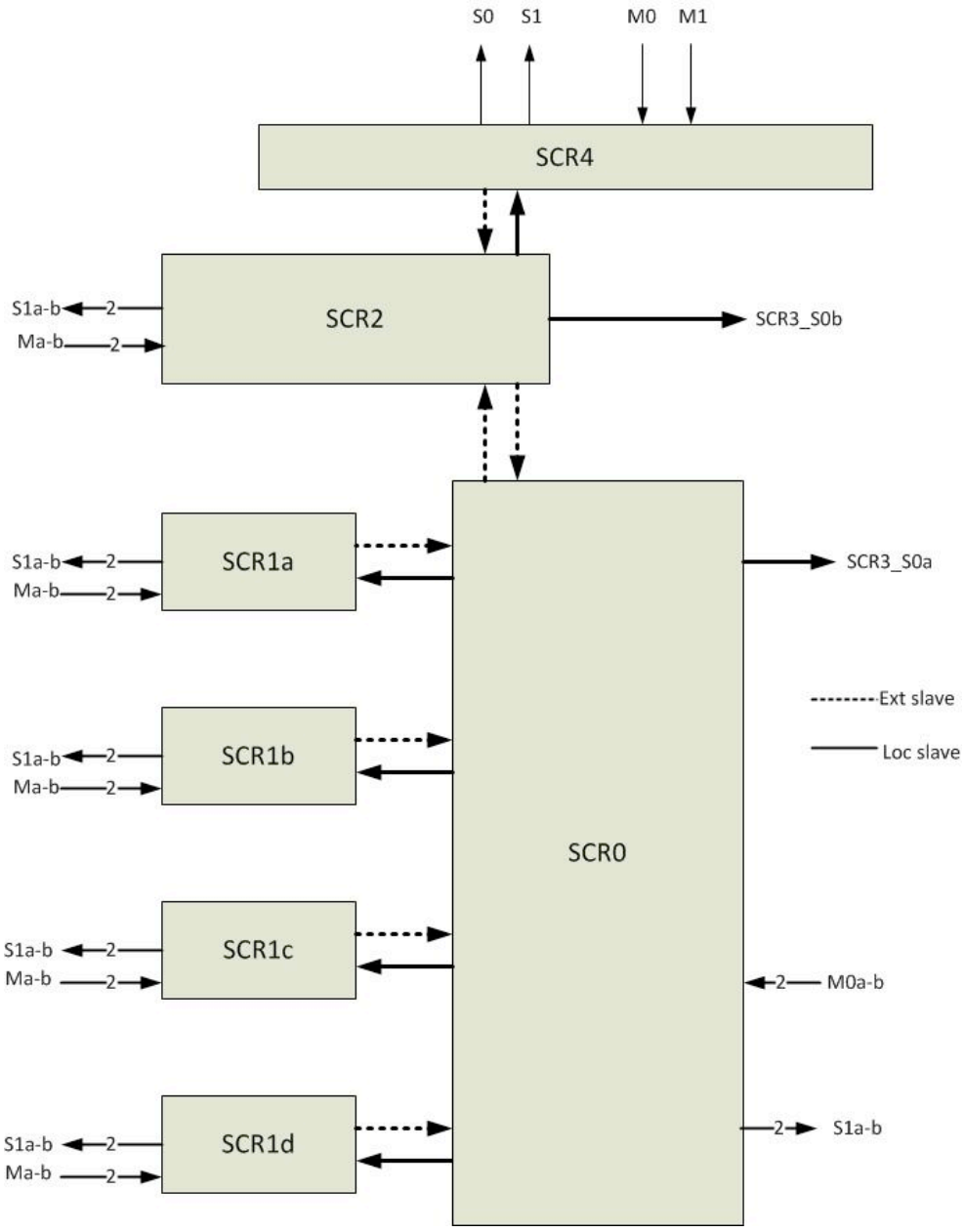

Figure 55: Configuration 4b- Interconnect for TI Based Subsystem

Analysis: No Deadlock

Conclusions

Overall, the effects of architectural modifications to Switching Central Resources (SCR) based NoCs have been provided, and these can be easily used for early estimations of Power, Performance and Area (PPA) of similar NoC designs. Honoring the valuable design cycle time of any design, such estimations provide a baseline and quick decision making platform. The dominant parameters, such as number of masters and slaves, arbitration techniques, pipeline depths and voltage supply level are prominent to any NoC design. The behavior of a SCR based NoC with variations of these parameters, have been illustrated. Such illustrations help designers to identify an architectural sweet spot to meet target specifications as shown in this report. The complete analysis has been discussed using a case study during the early design cycle phase of a Texas Instruments (TI) flagship product - wireless base station device.

As explained earlier, deadlocks are a major concern in any distributed NoC architectures, which use multiple issue bus protocols such as ARM-AXI and TI-VBUSM. Deadlocks, which occur in such NoC architectures, have been classified and analyzed in this report in detail for the first time. While resolving deadlocks, one may need to modify the architecture of interconnect effecting the PPA. Hence, a vicious circle between PPA estimation and deadlock resolution exists. The importance of deadlock identification has been discussed in this report. A proven algorithm to identify such deadlocks has been illustrated and verified with application to the Texas Instruments (TI) flagship product. This analysis and algorithm can be further extended to create a user-friendly tool to aid designers to make early architectural decisions.

References

[1] Texas Instruments Application Report SPRAAX6 August 2008, TMS320C6474 Common Bus Architecture Throughput.

[2] M J Flynn, W Luk, "Computer System Design: System on a Chip", Wiley 1st Edition

[3] ARM Inc., AMBA open Specifications

[4] H Kyung, G Park, J W Kwak, T Kim, S Park, "Design and implementation of Performance Analysis Unit (PAU) for AXI-based multi-core System on Chip", Microprocessors and Microsystems – Vol 34, Issue 2-4 March 2010

[5] K Gunther, "Prevention of Deadlocks in Packet Switched Data Transport Systems ", Communications IEEE transactions Jan 2003

[6] A Lankes, T Wild, A Herkersdorf, S Sonntag, H Reining," Comparison of Deadlock Recovery and Avoidance Mechanisms to Approach Message Dependent Deadlocks in On-chip Networks", NOCS'10

[7] Texas Instruments, "OMAP-L1x/C674x/AM1x SoC Architectural Overview"

[8] Arteris, "TI OMAP™ 4 Platform and Arteris Network-on-Chip IP"

[9] P Huang, W Hwang, "2-level FIFO architecture design for switch fabrics in network-on-chip", ISCAS 2006

[10] L Mhamdi, K Goossens, I V Senin, "Buffered Crossbar Fabrics based on Networks on Chip", IEEE CNSR 2010

[11] G Palermo, C Silvano, "PIRATE: A framework for power/performance exploration of Network on chip architectures", PATMOS 2004

[12] K Lee, S Lee, H Yoo, "A Distributed Crossbar Switch Scheduler for On-Chip Networks", (CICC 2003)

[13] A Agarwal, "Ram Computation," Scientific American, Vol. 281, No. 2 August 1999

[14] S Kumar, A Jantsch, J Soininen, M Forsell, M Millberg, J Oberg, K Tiensyrj, A Hemani, "A Network on Chip Architecture and Design Methodology," Proceedings of IEEE Computer Society Annual Symposium on VLSI, April 2002

[15] M Alho, J Nurmi, "Implementation for Interface Router IP for Proteo Network on Chip", Proceedings of the International Workshop on Design and Diagnostic of Electronic Circuits and Systems, 2003

[16] IBM, "CoreConnect Bus Architecture"

[17] B Vermeulen, J Dielissen, K Goossens, C Ciordas, “Bringing Communication Networks On Chip: Test and Verification Implications”, IEEE Communications magazine 2003

[18] P Martin, “A comparison of Network on Chip and Buses”, http://www.arteris.com/noc_whitepaper.pdf, 2005

[19] Arteris S A, “ From Bus and Crossbar to Network on a Chip”, 2009

[20] S Furber, J Bainbridge, “Future Trends in SoC interconnect”, Proc. IEEE Int’l Symp. on Design, Automation and Test (VLSI-TSA-DAT) 2005

[21] J Silc, B Robic, T Ungerer, “Processor Architecture: From Dataflow to Superscalar and Beyond”, Springer; 1st Edition

[22] J Henkel, W Wolf, S Chakrashar, “On-chip networks: A scalable, communication-centric embedded system design paradigm”, 17th International Conference on VLSI Design, 2004

[23] G D Micheli, L Benini, “Networks on Chips: Technology and Tools (Systems on Silicon)”, Morgan Kaufmann; 1st Edition

[24] OpenCores Organization “WISHBONE System-on-Chip Interconnection Architecture for portable IP Cores”, 2002

[25] OCP-IP , “Open Core Protocol Specification”, Release 2.1

[26] B Mathewson, “ Evolution of SoC Interconnect and How NoC fits in”, DAC 2010

[27] P P Pande, C Grecu, A Ivanov, R Saleh, “High-Throughput Switch-Based Interconnect for Future SoCs,” Proc. Third IEEE Int’l Workshop System-on-Chip for Real-Time Applications, 2003.

[28] C Minkenberg, F Abe, E Schiattarell, “Distributed Crossbar Schedulers” , IEEE workshop for High Performance Switching and Routing, 2006

[29] A Agarwal, C Iskander, R Shankar, “Survey of Network on Chip (NoC) Architectures & Contributions”, Journal of Engineering, Computing and Architecture, Volume 3, Issue 1, 2009

[30] M Ruggiero, F Angiolini, F Poletti, D Bertozzi, L Benini , R Zafalon, “Scalability analysis of evolving SoC interconnect protocols”, Int. Symp. on Systems-on-Chip, 2006

[31] Y Zhang, M Irwin, “Power and performance comparison of crossbars and buses as on-chip interconnect structures”, 1999

[32] A B Kahng, B Li, L Peh, K Samadi, “ORION 2.0: a fast and accurate NoC power and area model for early-stage design space exploration”, DATA’09, Proceedings of the Conference on Design, Automation and Test in Europe.